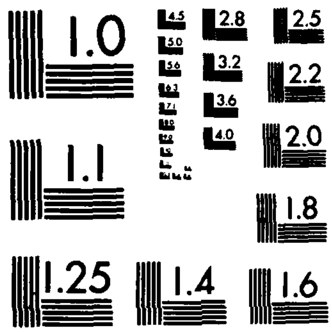
AD-A135 704 INTERACTIVE COMMUNICATION SYSTEMS SIMULATION MODEL (ICSSM) EXTENSION(U) HAZELTINE CORP GREENLAWN NY I GERRY ET AL. JUL 83 6479 RADC-TR-83-165 1/2

UNCLASSIFIED F30602-81-C-0001 F/G 17/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

RADC-TR-83-165
Final Technical Report
July 1983

INTERACTIVE COMMUNICATION SYSTEMS SIMULATION MODEL (ICSSM) EXTENSION

Hazeltine Corporation

Irene Gerry, Mary Mammone and William D. Wade

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-83-165 has been reviewed and is approved for publication.

APPROVED: Peter K. Leong

PETER K. LEONG
Project Engineer

APPROVED: Bruno Beek

BRUNO BEEK, Technical Director
Communications Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DCLF), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-165	2. GOVT ACCESSION NO. A135704	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE COMMUNICATION SYSTEMS SIMULATION MODEL (ICSSM) EXTENSION		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Jan 81 - July 82
		6. PERFORMING ORG. REPORT NUMBER 6479
7. AUTHOR(s) Irene Gerry Mary Mammone Dr. William Wade*		8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hazeltine Corporation Greenlawn NY 11740		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45192022
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (DCLF) Griffiss AFB NY 13441		12. REPORT DATE July 1983
		13. NUMBER OF PAGES 178
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Peter K. Leong (DCLF)

*Wade Engineering Company

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Point-to-point Communications Design & Analysis Aids
Communication Systems Modeling
Multi-port Block Diagram Modeling
Interactive Computer Simulation
Time-step/Event-step Simulation
Monte Carlo Simulation
User-System Interface
Modular Software Structure
Re-entrant Modules

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Interactive Communication System Simulation Model (ICSSM), developed for the Rome Air Development Center, is capable of simulating a point-to-point communication system including its functional elements, components, propagation effects, and transmission media. The ICSSM is a flexible, expandable, sophisticated and easy-to-use computerized means to develop or configure communication system specific simulation models; specify and validate system requirements; evaluate new techniques and assess the (over)

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

performance of existing and proposed conventional and ECCM communications systems and equipment.

The ICSSM's preconfigured programming structure frees the analyst from the burden of constructing a special simulation framework for each model effort, thus permitting him to concentrate on the model formulation itself. Further, the analyst may benefit from the legacy of previous modeling via the ICSSM library of communication model elements which are supported by computerized tutorials and guides.

The development of the initial ICSSM concentrated on efficient system sturcture, a generalized simulation capability and on making the system easy to use. The ICSSM increases in utility with continued use as additional modleing elements are incorporated into the expandable library.

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A/1	

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

ACKNOWLEDGEMENTS

Many have contributed with persistence, skill and imagination to the success of the ICSSM System Extension project. In particular, Hazeltine Corporation was pleased to have, throughout this project, the constant support, able guidance, and technical assistance of P. Leong and D. McAuliffe of the Rome Air Development Center of USAF. Credit is due I. Gerry, G. Gavey, D. Ciccolella, M. Mammone, K. Mester Rossetti, S. Arbeeny and D. Schuster of Hazeltine Corporation, and ' Wade of Wade Engineering Company, who, with the encourageme of W. Griese, Head of Hazeltine's Systems Analysis and Ev -ation Laboratory, and R. H. Cope, Vice President for R search, Hazeltine Corporation, successfully completed developments reported on here.

SECTION I

IMPETUS AND BACKGROUND FOR ICSSM DEVELOPMENT

1.1 PURPOSE

This document constitutes the Final Technical Report (FTR) on the Interactive Communication System Simulation Model (ICSSM) developed by Hazeltine Corporation for the Rome Air Development Center under Contract No. F30602-78-C-0197 and F30602-81-C-0001.

The purpose of this FTR is to summarize the results of ICSSM development. It is organized into five sections that describe the motivations, rationale, implementation, results, and recommendations for the ICSSM system.

This section provides a sketch of the background and history of the ICSSM development project and outlines the functional requirements and design objectives of the ICSSM system.

1.2 SPONSORSHIP AND PROJECT GUIDANCE

ICSSM system development was undertaken in response to requirements described in Request for Proposal F30602-77-R-0185 (PR No. C-7-2027) and PR No. C-0-2035. These requests were responded to by Hazeltine Corporation via "Technical Proposal for Interactive Communications System Simulation Model," Hazeltine Report No. 6326, dated December 14, 1977, and by "Technical Proposal for Interactive Communication System Simulation Model Extension," Hazeltine Report No. 6403R1, dated August 22, 1980.

The ICSSM was designed and developed under the guidance of RADC/DCLF personnel, in accordance with the items of description in the Statement of Work for PR No. C-7-2027 and PR No. C-0-2035. The development, test, and documentation of the ICSSM are in accord with extant, applicable Air Force and DOD guidelines, standards, and procedures, including:

- RADC Computer Software Specification CP07877796100E
- DOD Manual 7935.1-S for Automated Data Systems Documentation Standards

Programs for the Extended ICSSM system and its validation models were developed and tested on RADC's Honeywell 6180 computer operating under the Multics Software Release 7.0. The Honeywell ANSI-66 compiler was used.

Test procedures and results exhibited in this report are demonstrable when the Extended ICSSM is executed using the computer facilities described above.

1.3 REFERENCES

Relevant technical references that define, express or provide technical details of specific applications models that demonstrate ICSSM operation or set the tone for some aspects of ICSSM design are included in Appendix A. Hazeltine documents previously published in connection with the ICSSM project are also listed in Appendix A.

1.4 ICSSM SYSTEM DESCRIPTION AND CAPABILITIES

The ICSSM system consists of a software executive or control program, model specification and data reduction programs, plus an Applications Library of computerized modeling elements for non-real-time computer simulation of point-to-point digital communication systems. The ICSSM provides U.S. Air Force engineers and scientists with a powerful, simple-to-use, interactive simulation capability operating on a computer system at RADC.

1.5 BACKGROUND AND MOTIVATION FOR ICSSM DEVELOPMENT

The application of digital computer simulation to communication systems analysis and design has a long history. The efficacy of these efforts depends upon the software and hardware available at the time of their conception and execution, upon the state of the software engineering art, and upon the sophistication of the communication analyst's science and art at the time. Many of those simulation efforts could be said to have been successful, particularly when the then current states of computer science and communication science are considered. However, the recent literature and experience strongly suggest the overall characteristics of an improved communications system simulator:

- Sophisticated wideband and computation-intensive communication system designs are of increasing interest. A simulator should operate at <u>high speed</u> to deal with the very large number of data samples that may be needed to characterize the operation of such communications systems.

- o Anticipated advances in the communication systems art require that a simulator be flexible and adaptable if it is t service the analyst's future needs.

- o The sophistication of modern communication systems and the emphasis placed on performance prediction during their design phase requires that a simulator be accurate.

- o The rapid growth of communication science and the need both to express new analytic and design ideas and to explore the performance limits of communication system configurations operating under very adverse conditions require that a simulator be expressly general in its application potential and to possess an extraordinary degree of faithfulness and validity in its results.

- o The increasing specialization apparent in all fields of technical endeavor, including the computer and communication sciences, requires that a simulator be unobtrusive and easy to use, with apparent relevance to the communications analyst's view of the problem.

- o The simulation should take advantage of new and anticipated capabilities in available computer equipment and operating software if it is to remain viable. Therefore, the simulator must be transportable, using generalized computational technique.

- o The number of functioning communication/processing elements of the communications systems to be simulated and the level of detail needed to explore their performance are very variable. This can result in very large configurations or collections of computational or functional elements within a single model. The effective simulator must be elastic or expandable in configuration and simulation capability.

Thus, the requirements for a state-of-the-art communication system simulator are defined by the properties of:

- o Speed
- o Flexibility/Expandability/Adaptability
- o Generality
- o Accuracy/Validity

o Ease of Use/Relevance

o Transportability

1.6 DESIGN OBJECTIVES FOR ICSSM

The design objectives for the entry-level ICSSM configuration are related directly to the requirements/properties disclosed below.

1.6.1 Speed as an Objective

Software overhead attributable to ICSSM control program operation should be less than 10% of the computational load. Simulation-model-specific process requirements should account for at least 90% of processing time for simulations. When host hardware is augmented by an array processor, the ICSSM should provide a processing ratio of 1 second real time to 1 hour machine time for a selected system model of current interest, chosen for the detail of its representation and for its concomitant heavy computational burden.

1.6.2 Accuracy and Validity as Objectives

The ICSSM should be designed so that the validity and accuracy of the results of a modeling exercise are dependent solely on the faithfulness with which the user designs and implements the functional elements of the model, and not on the internal structure of ICSSM control/executive software.

1.6.3 Ease of Use and Relevance as Objectives

ICSSM design should emphasize the natural analytic and mathematical constructs of the communications analyst. It should provide (interactive) input facilities couched in language and familiar-in-form or relatable to the conceptual needs of the analyst or communications engineer.

The ICSSM should further provide a simple regimen for describing (and, where possible or appropriate, altering previously described) communications models, in computer-efficient form, with no unnecessary artifices that impede the analyst's intuitive "feel" for the model.

1.6.4 Transportability as an Objective

The ICSSM should be designed in the standard (ANSI) FORTRAN language and should avoid reliance on computer operating system capabilities or resources for its internal operation. Nor should it rely on peculiarities or special features of

particular computers to meet its objectives, unless these special features can normally be expected to exist on the computer systems likely or intended to provide host for the ICSSM.

1.7 IMPROVEMENTS AFFORDED BY ICSSM

ICSSM design capabilities are directly related to user needs. The requirements envelope of paragraph 1.7 et seq defines a simulation capability that improves significantly over some existing simulation facilities, constituting a unique and novel confluence of desirable features. In brief, the improvements afforded by the ICSSM are:

a. Accuracy and applicability limited only by the ingenuity and modeling requirements of the user and by the capacities of the host computer.

b. Flexibility and adaptability that accommodates common design/analytic regimens in communications system engineering and naturally suggest or accommodate other regimens, some heretofore too impractical for common implementation.

c. Ability to model and examine portions of communication system designs in greater or lesser detail, completely in consonance with the user's actual interest and needs.

d. Ability to decompose a communication system model into convenient "pieces" that can be exercised sequentially and repeatedly, each time using the entire host computer resource.

e. Natural accretion and reuse of functional electronic modeling elements (via registration in the permanent ICSSM Library) so that the legacy of previous modeling efforts need not be lost.

f. Facilities for selecting and controlling the types and volume of output data generated and collected in accord only with the user's particular interests and needs.

g. Interactive facilities that preserve the user's "feel" for his model and render the presence of the simulation software unobtrusive, except where discipline is imposed to ensure model correctness and efficient use of host computer resources.

h. Simulation time (end-to-end computer processing time) determined almost exclusively by the physical nature of the communication system being modeled, the detail with which

the user wishes to express the model and examine its performance, and the throughput capabilities of the host computer, and almost independently of the internal structure and requirements of the ICSSM control/executive software.

1.8 LIMITATIONS PRESENT IN ICSSM DESIGN

The generality and flexibility of ICSSM design and its emphasis on transparency to the user certainly relieve many limitations now present in other simulation methods. Specific limitations arise or are imposed by host computer resources and by structural disciplines needed for Library element design. The ICSSM is designed for point-to-point digital communication system simulation. However, continuous waveform system simulation (eg, an analog FM system) can be handled but only in sampled-data form.

It is possible to exceed current ICSSM capabilities if simulations are attempted that effectively require more samples (computer data elements) within an ICSSM internally-defined sample/event packet than the ICSSM can represent and still remain true to physical law (eg, it is possible to violate the sampling theorem).

When simulation study of the extremely rare event is required (eg, extremely small probability-of-error operating points), the basic random process for simulating noise corruption, for example, may never be exactly suitable in that it will never be exactly random, no matter what statistic is of interest. Thus, it is possible to define a simulation that will never produce enough valid samples before termination of the simulation.

The ICSSM has been designed so that it may be used with an Array Processor of the capabilities of the Floating Point Systems, Inc., Model AP120B. The Array Processor can increase the speed of some ICSSM simulation executions. The ICSSM possesses software that emulates AP120B Array Processor operations (these are available in the Applications Library). However, using these emulation facilities will adversely affect the speed of the simulation exercises themselves.

SECTION II

ICSSM DESIGN RATIONALE

2.1 INTRODUCTION

This section describes the rationale for ICSSM design. It addresses the modeling philosophy, the communications system representation problem, the principles of validation, and the key design features that help determine the ICSSM system configuration.

2.2 THE BLOCK DIAGRAM APPROACH TO COMMUNICATIONS SYSTEM MODELING

ICSSM design is based upon a "block diagram" approach to communications systems modeling. This approach is epitomized in figure 2-1. There, elementary functional processes employed in a communications system are represented by blocks. Each block represents a separate (mathematical) function that maps elements of a domain space (input port) into elements of a range space (output port). The sequence with which the functions are applied is indicated by connecting lines drawn from an output port of a function block to an input port of some (usually) other function block. The connections themselves represent mathematical identity operators.

The function blocks represent unilateral operations; while they may represent mathematically invertable operators, they are never considered in this light. Thus, in this approach, the inverse of a given function, if required, is always represented by a block (or blocks) representing the inverse per se.

2.2.1 Domains and Ranges in ICSSM Models

The elementary functional processes represented on figure 2-1 map particular domains into particular ranges. For example, a function may map a set of time-dependent voltage values (domain) into another set of time-dependent voltage values (range). Within the ICSSM system, the user has complete freedom to define the "meaning" of any domain or range space consistent with his modeling requirements. Particular elements (which could in themselves be functions) from the desired domain or range sets are represented within the ICSSM as (ordered) sequences of values of the dependent variable required. The nature of the corresponding (ordered)

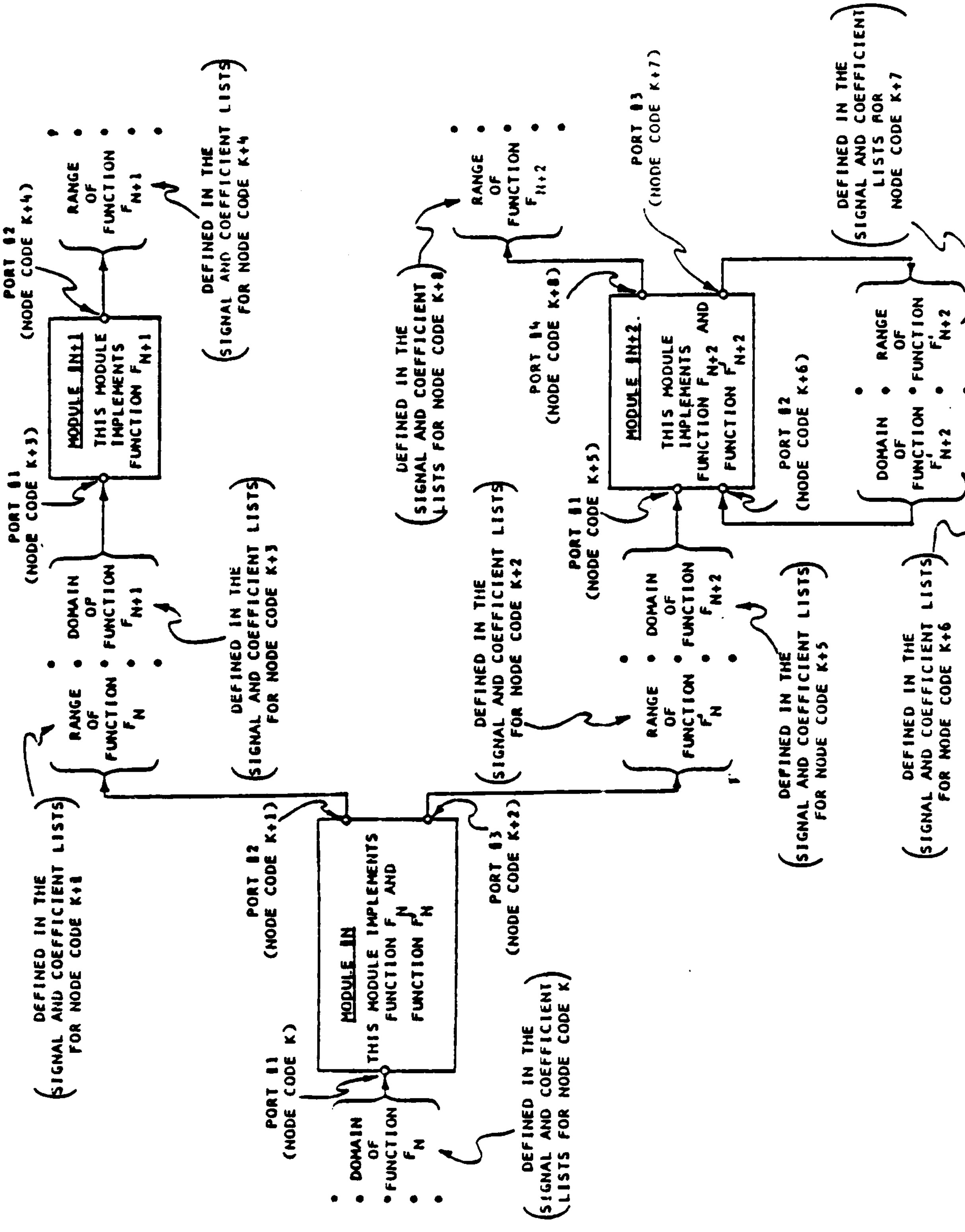


Figure 2-1. Communications System Modeling - Block Diagram Approach

sequence of values of the independent variable are implied or defined by the nature of the associated elementary functional process.

For example, suppose a domain were to consist of (time-ordered, regularly-spaced) amplitude samples of a function representing a signal voltage. The domain would then be represented in the ICSSM as a set of values (called Coefficients) arranged in time-ordered sequence and made available as input to the relevant elementary functional process manifested in a related ICSSM Applications Library module. The results of the functional processing would be generated (by the actions of the Applications Library module/algorithm) as an ordered sequence of range values made available within the ICSSM for subsequent processing by other modules.

2.2.2 Coefficient and Signal List Records in ICSSM

Within the ICSSM, ordered sets of domain and range values are termed Coefficient Records and are managed within the ICSSM as Coefficient Lists. The Coefficient Records associated with a particular port of a particular module of the block diagram are identified within the ICSSM system by a combination of module number and port number from which a unique Node Code number is derived. The definition or meaning ascribed to the elements of particular Coefficient Records depends on the associated Applications Library module selected or specified by the user. The generation of Coefficient Records and Node Codes, and the arrangement and control of the Coefficient Lists is transparent to the user.

Associated with a given domain or range set (ie, Coefficient Records), the ICSSM employs Signal List Records containing values of descriptors, attributes, derived quantities, or describing/delineating quantities related to the associated functional module or Coefficient Record.

For example, if a Coefficient Record were to contain values of power associated with particular frequencies in the spectrum existing at a particular Node, the Signal List Record would perhaps contain values for the frequency spacing of the spectral values and perhaps a measure of the total extent in frequency (ie, the bandwidth) represented by the Coefficient Record. Generally, associated with a Coefficient Record, one may need or find additional Signal List Record quantities. Some of the Signal List elements are "derived from" the Coefficient Record elements by integral operations (eg, a Signal List value of average power found associated with Coefficient Record entries representing

voltage as a function of time). Sets of values constituting a Signal List Record are also identified by a Node Code number as with Coefficient Records. Management and manipulation of Signal List Records are transparent to the user.

2.2.3 Generality and Flexibility in ICSSM

The ICSSM achieves generality and flexibility in its models in part through the use of the aforementioned Coefficient Record and Signal List Record concepts. Concomitantly, some caution is called for in algorithm design for ICSSM Applications Library modules and in actual model formulation using the ICSSM. Care must be taken to confirm that the measurement units of the output (range) Coefficient Record and Signal List Record elements for a given module are compatible with the units required by the input port (domain) units of the module to which it may be connected. If, for example, a given module were to produce a spectrum in its output Coefficient Record and it was connected to another module that used amplitude values of a bit stream in its input Coefficient Record, then these modules would be incompatible. However, the ICSSM would not warn the user of such an impending incompatibility during the ICSSM-aided model configuration steps. Such caution must be taken by the user so that he knows the nature of the modules he seeks to interconnect in his model.

For some models, simulation may be performed without requiring Coefficient Records and Lists at all. In such cases, the algorithmic forms for the Applications Library modules constituting the model address more general properties or attributes of simulated communications signals, and not the signals themselves. By this means, models may be constructed that operate at levels of generality or abstraction other than that represented by bit-by-bit or signal-sample-by-signal-sample processing.

2.3 REPRESENTATIONAL AND ANALYTIC PROBLEMS IN COMMUNICATIONS SYSTEM MODELING

ICSSM design is strongly influenced by the current and conceivable future states of the communication analyst's art. A brief discussion of the relevant issues will clarify the role of simulation in the practice of this art, and it will isolate those themes that most strongly impact ICSSM design, particularly with regard to the design of functional elements (ie, Applications Library modules) employed in the ICSSM.

The communication modeler's implicit assumption is that the mathematical framework of the communication art is somehow bound. This assumption may be justified on several bases. For example, we may consider that all functions (ie, signal representations) used in the art are square-integrable (Lebesque sense). Thus, all operators on such functions must map the space L_2 into L_2. This already restricts the communication theorist's consideration substantially. Further, all functions suitable for a simulation representation must be computable (Davis), thus, further restricting the class of functions to be considered. These and similar mathematical assertions may be too general for practical use here, but they do provide general assurances. Yet, a large body of electronic practice already exists that, from another point of view, also allows the assertion that the communication design discipline is far more bound than the literature might indicate. To justify this claim, it is useful to briefly examine one particular class of problems. This example class will also motivate some general observations about the nature of the overall modeling and simulation problems for the communication analyst.

2.3.1 A Problem Example

The selected examples are characterized by the presence of additive interference constituting a sum of sinusoids - the multiple sinusoid interference (MSI) case. The central limit theorem does not apply here so that a Gaussian assumption is invalid. Instances of this class include: co-channel interference dominated by a few (non-spread-spectrum) signals; intersymbol interference; self-interference in a frequency-time hopped (FTH) CDMA regime; and "multiple continuous wave" jamming.

For the most part, the literature (eg, ref 9-21) has dealt with this interference type in the context of receiver designs optimum only for additive white Gaussian noise (AWGN). The various analyses can be divided into two approaches: numerical methods (ref 9-15) and bounding approaches (ref 16-21).

The numerical methods include series methods, the Gaussian quadrature method, and the direct averaging method. When the vector of random variables representing the MSI has certain properties (which usually occur in practice), all these methods converge. However, the pertinent expressions have not been evaluated exactly. Truncations seem unavoidable, but for each method, the consequent errors are bound and vanish in the limit. Yet, simple and accurate expressions seem practically impossible to obtain. No

ranking or general critical comparison is available for these methods. Unfortunately, the current numerical methods do not consider many practical effects (eg, the influence of the MSI on carrier and timing recovery or the presence of system non-linearities).

The bounding approaches trade accuracy for a reduction in analysis complexity. These approaches fall into two classes: those using the Chernoff bound and those using the Maximizing Distribution bound. Results based on the Chernoff bound grow more accurate (ref 16-18) as the signal-to-interference ratio increases. Perhaps the greatest virtue of the Chernoff bound is the insight it furnishes as to when it is reasonable to treat MSI as equivalent to AWGN of the same average power.

The Maximizing Distribution bound has been a most successful MSI analysis approach. The results obtained (ref 19-21) permit families of curves to be developed and stored as a table. The bound is tight enough for most cases of interest. However, the same practical effects are ignored as in the numerical methods. Thus, for more detailed works, a regime using a combination of analysis and simulation computation is required.

Very little (ref 22, 23) has been accomplished in identifying the form of the optimum receiver when both MSI and AWGN are present. The meager results ignore multiplicative interference and other practical effects.

2.3.2 The Status of Communications Analytic Theory

The above example illustrates characteristics of the communications discipline in general:

a. The literature consists primarily of alternative analyses for a limited set of basic problems.

b. Typically, one of the alternatives provides the best approach; other alternatives merely augment the insights derived from the numerically best approach.

c. Invariably, the analyses fail to treat many of the factors relevant to the real problem from which the modeled problem is abstracted.

d. The literature rarely addresses the derivation of structures optimum for real problems. With few exceptions, those constructs employed are optimum for Gaussian statistics.

e. At present, communication system design is based on Gaussian-optimum constructs. The non-Gaussian constituents are then accommodated by a combination of adaptive processing and ad hoc design.

Point a, above, implies that the discipline of communication systems is relatively well-bound, an attribute of fundamental importance to the viability of ICSSM Applications Library concept.

Point b reinforces point a and emphasizes a major merit of the ICSSM: the common framework permits alternative design approaches to be simulated and compared so that inferior designs can be discarded.

Point c, d and e reflects the remarkably limited power of "applied mathematics" in dealing with the complex problems that arise in modern communication systems. Communication system design will ignore, in most cases, the goal of completely analytical solutions, in favor of simulation as a major analytic/computation tool.

2.3.3 Merits and Potentials of Simulation

A fundamental merit of the ICSSM is that, by assuming much of the computational burden, it permits the communication theorist/analyst to devote more time and ingenuity to correcting the pervasiveness of the Gaussian assumption, as cited in point d.

Consider, for example, the block diagram of figure 2-2 which describes a broad class of communications link designs. If the application is digital communications (as in DCS-LOS), the modulation is most likely based on a signal-space geometry that minimizes (consistent with channel bandwidth limitations) the cross-correlation among an M'ary signal set. This approach is optimum for the AWGN channel model but is generally sub-optimum for colored noise. For non-Gaussian statistics, it is patently sub-optimum.

Consistent with the Gaussian noise premise, the demodulation probably embodies matched filtering or correlation to maximize the signal-to-noise ratio of the decision statistic.

Similarly, for analog parameter communication (eg, Pseudo-Noise Conferencing Modem), the demodulation invariably is based on a Gaussian assumption for the interference random processes. The applicable science in this case is estimation theory wherein the statistics for both the information process and interference are relevant.

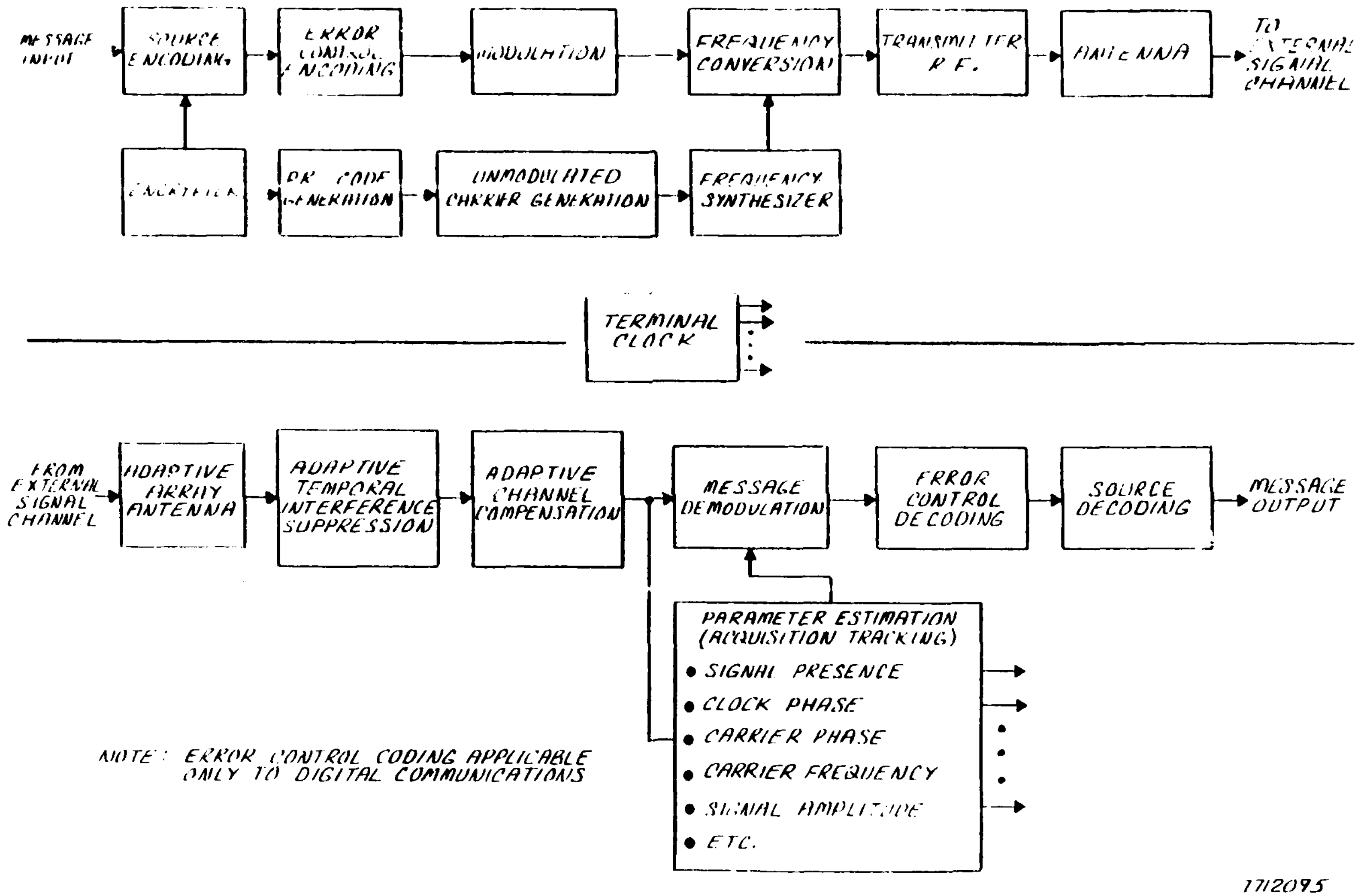


Figure 2-2. Block Diagram for Illustrative Communication System

The fidelity criterion selected (primarily because of tractability) is either the minimization of the mean-square estimation error (MME) or the minimization of the error variance for an unbiased estimate.

A state variable formulation of the problem permits the introduction of continuous Markov processes, but tractability invariably requires the assumption of a Gaussian-Markov process and usually dictates a linearization of the problem. This leads to the use of phase and frequency locked loops, which are (essentially) a linearization of the Kalman filter concept, itself optimum for linear estimation of a Gaussian process in Gaussian noise interference. These considerations carry over intact to the estimations of phase, frequency, and time in digital demodulation, as in figure 2-2.

The limited number of constructs available to the communication link designer promotes the manageability of the ICSSM Applications Library concept. By a thorough modeling of relatively few generic constructs, a very large portion of the modeling required in communication system design in the foreseeable future can be addressed.

2.3.4 Aspects of Communication Model Evaluation

The relatively few design constructs provided by the theory for communication link design derives from the intractability of expressions for optimum designs.

Analytical tractability also limits the complexity of performance criteria. To a great extent, this derives from inherent limitations of the underlying decision theory. In practice, decision criteria reduce to a few basic theories (eg, "minimization of symbol error probability," in digital communications, and "unbiased minimum error variance," for analog communications).

The designer often force-fits the aforementioned simple criteria to the actual problem. For a digital signaling design (as in DCS-LOS), this "force-fitting" manifests itself in the artifice of source encoding voice waveform into a sequence of "information bits." Consequently, the quality of input to the source decoder is judged in terms of bit-error statistics.

Typically, digital communication design is based on the theoretical criterion of minimizing the probability of bit error, which is merely one of an infinite number of statistics characterizing bit-error patterns.

Simulation of the system may permit the study of bit-error statistics other than bit-error probability and may reveal the existence of channel memory. If this memory effect degrades actual performance, subsequent design iterations may include elements to counteract it. Simulation may disclose that the actual clustering of bit errors permits a relaxation on the required bit-error probability (eg, speech intelligibility is known (ref 24, 25) to be highly resistant to bit-error bursts of certain durations).

In both digital and analog communication system design, unexpected or "anamolous" performance degradation is often traceable to higher-order error statistics (ie, other than the average bit-error rate of the average power of the voice-waveform estimation error). Significant degradation (if not outright failure) results from design concepts that apply simple performance criteria and idealized channel models to complex source/sink requirements and real channel characteristics. The anamolous behavior invariably involves statistics other than that chosen for optimization in the theoretical design.

Analyses that consider merely the theoretically optimum criteria often spawn designs that succeed on paper and fail in operation. Simulations can uncover the unexpected sources of degradation so that they can be properly mitigated in follow-up design iterations. Accordingly, the ICSSM system is structured to permit access to any information node in the link model formulation, and not merely to preselected performance measures.

2.4 ACCURACY AND VALIDITY IN SIMULATION

Ideally, the accuracy of mathematical calculation in a simulation should be limited only by the word size, internal coding, and arithmetic capability of the host computer.

The validity of simulation results should depend only upon the validity of the functional elements used within the formulated model. This, in turn, should depend upon: (a) the availability of algorithmic representations of the modeled processes; (b) the ingenuity and skill of the user in defining the performance limits and computational analogs manifested in simulation elements; and (c) the nature of the validation principles chosen.

Validation principles can be chosen from among several possibilities. Figure 2-3 shows an abstract representation of the conceivable modeling processes that need consideration. There, the relationships among the "versions of reality"

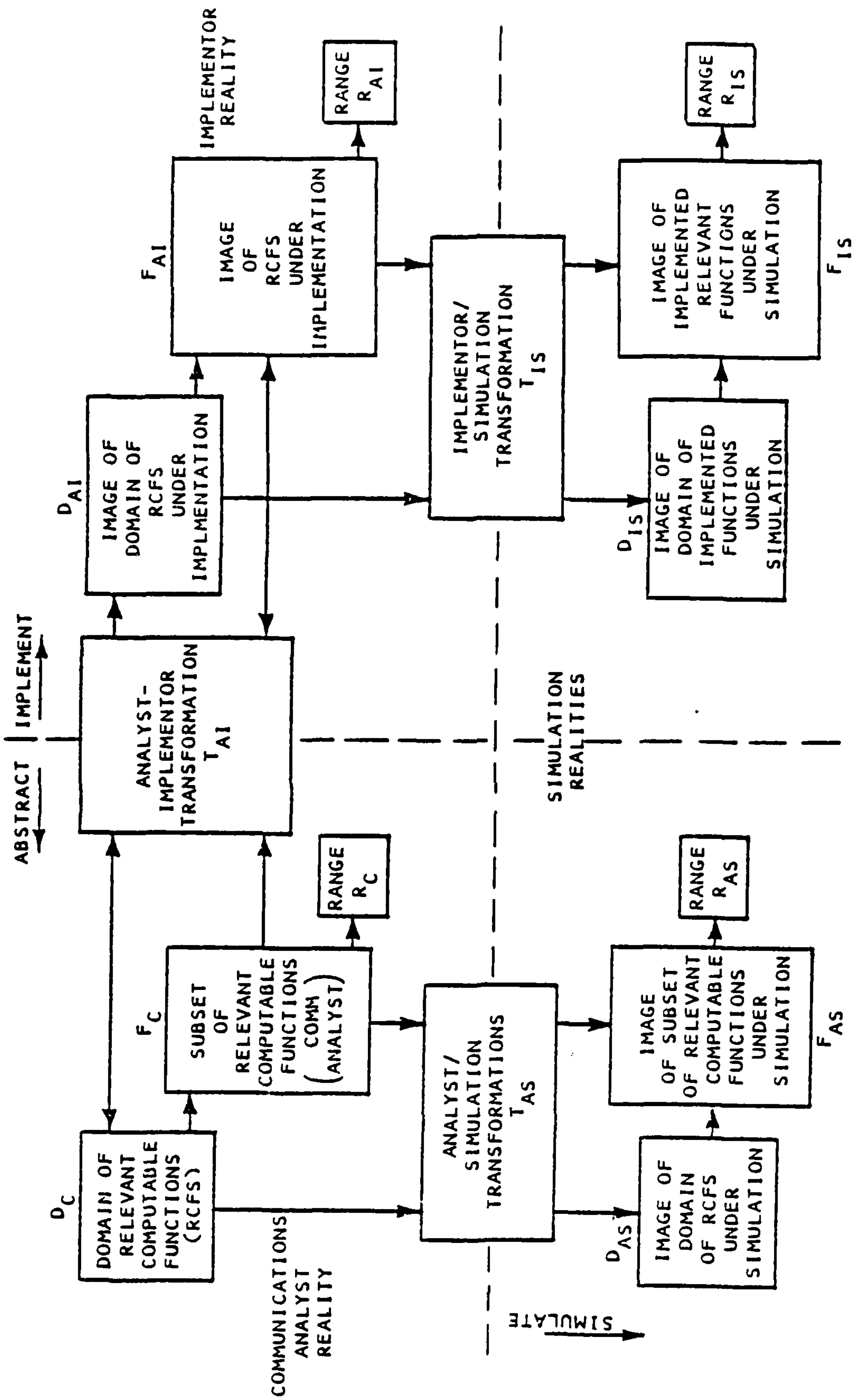


Figure 2-3. Versions of Reality

that exist in the communications system/simulation system setting are depicted. Validation consists of determining the fidelity of any of the transformation systems T_{AI}, T_{AS}, or T_{IS}.

The "fidelity" can be measured by determining how "close" a certain result in one of the output range spaces lies with respect to its range pre-image under the given transformation, given that the image of a corresponding domain element in the output domain space lies specifically close to its domain pre-image. Figure 2-4 portrays the described simulation modes as they bear on the fidelity relations. There, validation is perceived and portrayed as "comparisons" between domain/range pairs in one reality and the corresponding pairs in another reality. Validity is also determined by the metrics or "measures of closeness" chosen.

A simulator should be designed to be independent of both the metrics used and the simulation mode selected. Validity must then address the fidelity of modeling correspondence. The simulator's internal control/executive software should provide a facility to effect the mappings T_{AS} or T_{IS} but should be neutral with respect to their validity.

2.5 USING ICSSM FOR COMMUNICATIONS SYSTEM SIMULATION

The ICSSM system can simulate the performance of multiple-element communications systems. An ICSSM "target-simulation model" (TSM) can be fashioned to incorporate all the traditional elements of point-to-point links in one model: message source, coding and modulation steps; antenna, propagation and channel processes; receiver front end; demodulation and decoding processes; and message transduction. However, implementation limitations can restrict the number of different Applications Library modules used in a TSM. The number of functional elements that can be used, the number of interconnections permitted, and the time available for exercising a TSM all have limits. However, the ICSSM is capable of alternative styles of usage that can surmount most of the model-size limitations. Alternative methods of using the ICSSM are discussed in the paragraphs below.

2.5.1 Basic Method of ICSSM Use

The ICSSM consists of three main computer-based elements: the pre-simulation "Configurator" (MC Subsystem); the simulation model Exercisor (ES Subsystem); and the post-simulation "Output Processor" (PP Subsystem). These elements are supported by the ICSSM Applications Library and by

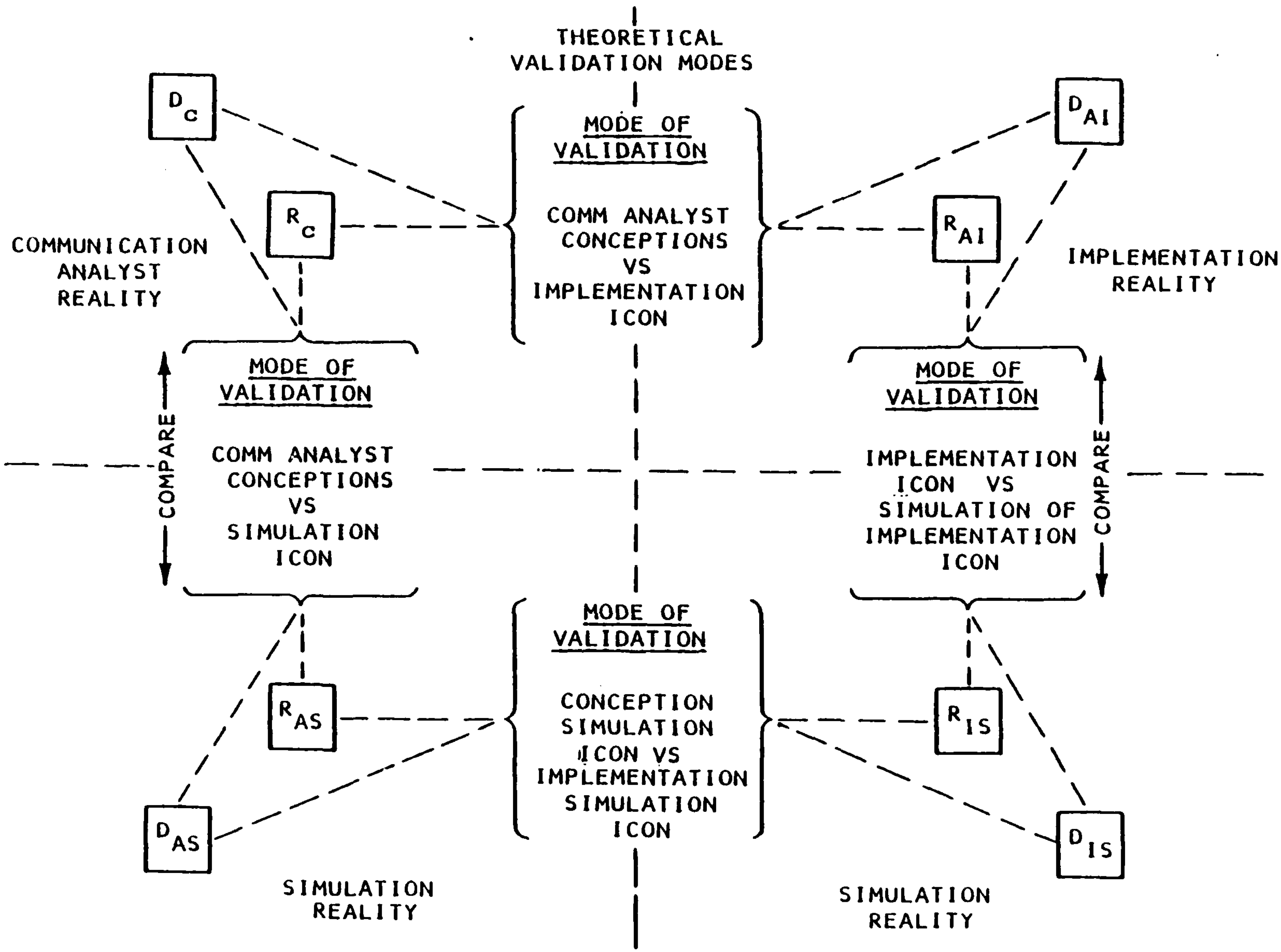


Figure 2-4. Theoretical Validation Modes

an auxiliary processor specifically designed for ICSSM Applications Library maintenance.

The ICSSM Applications Library contains many functional modules (sub-programs) that can be used in communication model construction. Access to this collection of modules is provided through a Library Directory and a Module Description File used mainly during Model Configuration. The selected functional modules (eg, envelope detector, signal generator, encoder, etc) are incorporated into the TSM during a pre-compile step. The ICSSM Applications Library also contains modules and program elements that simulate certain electronic test equipment and certain communication system testing/evaluation methods (eg, a module that computes bit-error-rate). TSM output data display/reduction processes are performed in the output data processing step either through data reduction programs and subroutines, which are available from the ICSSM Applications Library, or through plotting, printing, display, and processing software from the host computer's standard support software.

The basic method of ICSSM usage involves specifying/configuring the entire communications system model, submitting the resultant TSM for execution, and examining the resultant output data.

2.5.2 Iterative or Closed-Loop Use of ICSSM

The diagram of figure 2-5 suggests that the user employs ICSSM to complete a closed loop of methods, manipulations, and actions. This employment style can be extended to repeated or iterative use of the ICSSM facilities. This is consonant with the design or analysis of advanced communications systems since mathematical/analytic methods are often inadequate to characterize a system completely or precisely (refer to paragraph 2.3). The ICSSM is designed to accommodate the iterative method: TSM configuration specifications created by the (interactive) cooperation among the user and the ICSSM configuration program, and the ICSSM Library Directory "help" files are captured and formatted in an intermediate file (refer to Section III for more detailed description). The intermediate file contains a compacted description of the user modeling conception. This file may be retained and made accessible to the user using the ICSSM configuration program. Adjustment and reconfiguration of the user's model can thus be accomplished by modifying a previously configured TSM through the modification function of the model configurator selection program.

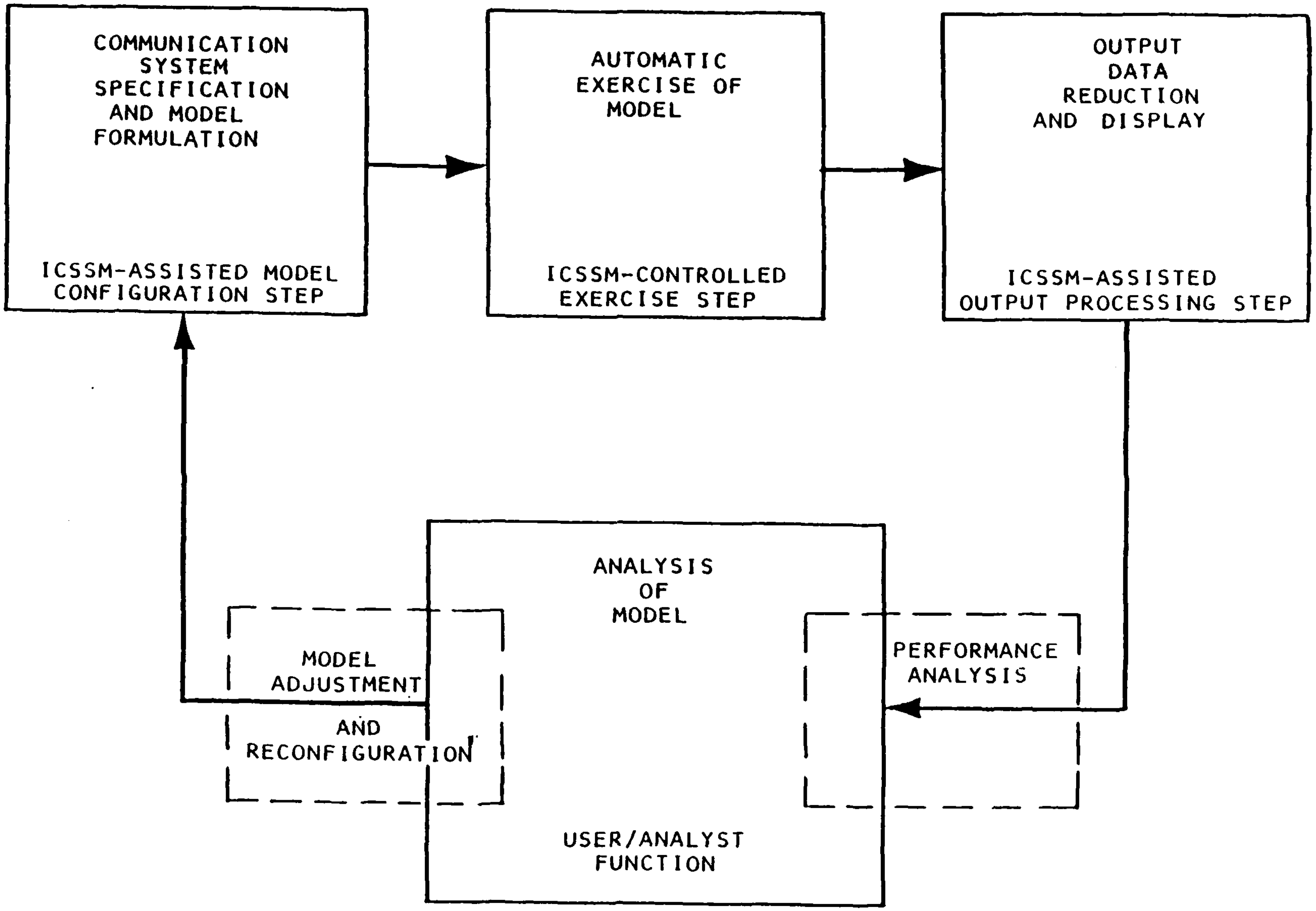


Figure 2-5. ICSSM's Role in Design and Analysis of a Communication System

By modifying a previously configured TSM via modification of the configuration specifications of the relevant intermediate file, the user can make adjustments in the TSM, submit the modified TSM specifications for compilation, and exercise the modified TSM. This process may be performed iteratively, saving the user effort otherwise involved in completely redefining the TSM for each iteration.

2.5.3 Partial Model/Experimental Data Use of ICSSM

Circumstances may be that a communications system to be modeled is so "large" or elaborate that it is convenient to "break it up" into segments or partial models. One convenient place to break up a model is at a natural interface (say, at the interface between that portion of the model representing the propagation channel/medium and that representing the receiver antenna). For example, the transmitter/channel/medium portion of a TSM is configured and exercised as an independent TSM (sub-model) under ICSSM control. Exercising this sub-model results in the collection of data simulating the effects of propagation anomalies (eg, multipath); medium disturbances (eg, dispersion); and jamming and noise on the simulated communication transmission. The data so collected is then formatted for use as an input data stream to a second, independent sub-model of the receiver antenna/demod/message transduction portion of the communication system, this portion having its own TSM configured and operating under ICSSM control. The "sizes" of the sub-models (there could be more than the two described in the foregoing example) would be smaller than the entire original TSM and thus more easily matched to the limitations imposed by host computer capacities and by computer operational considerations. Figure 2-6 depicts ICSSM use under the "partial models" method.

The "partial models" method of ICSSM use is particularly relevant to some of the common requirements in communications systems analysis/design. For example, a frequent analysis task addresses the comparison of the performance of several alternative designs under identical conditions. The ICSSM, when applied in this context, may be used to construct a sub-model configured to derive realistic transmission/distortion/disturbance data characterizing a given communication channel. The data is collected and applied successively to separate sub-models configured to model the alternative receiver designs. The output data obtained from the several receiver simulations can be compared directly, since they are obtained under the identical simulated

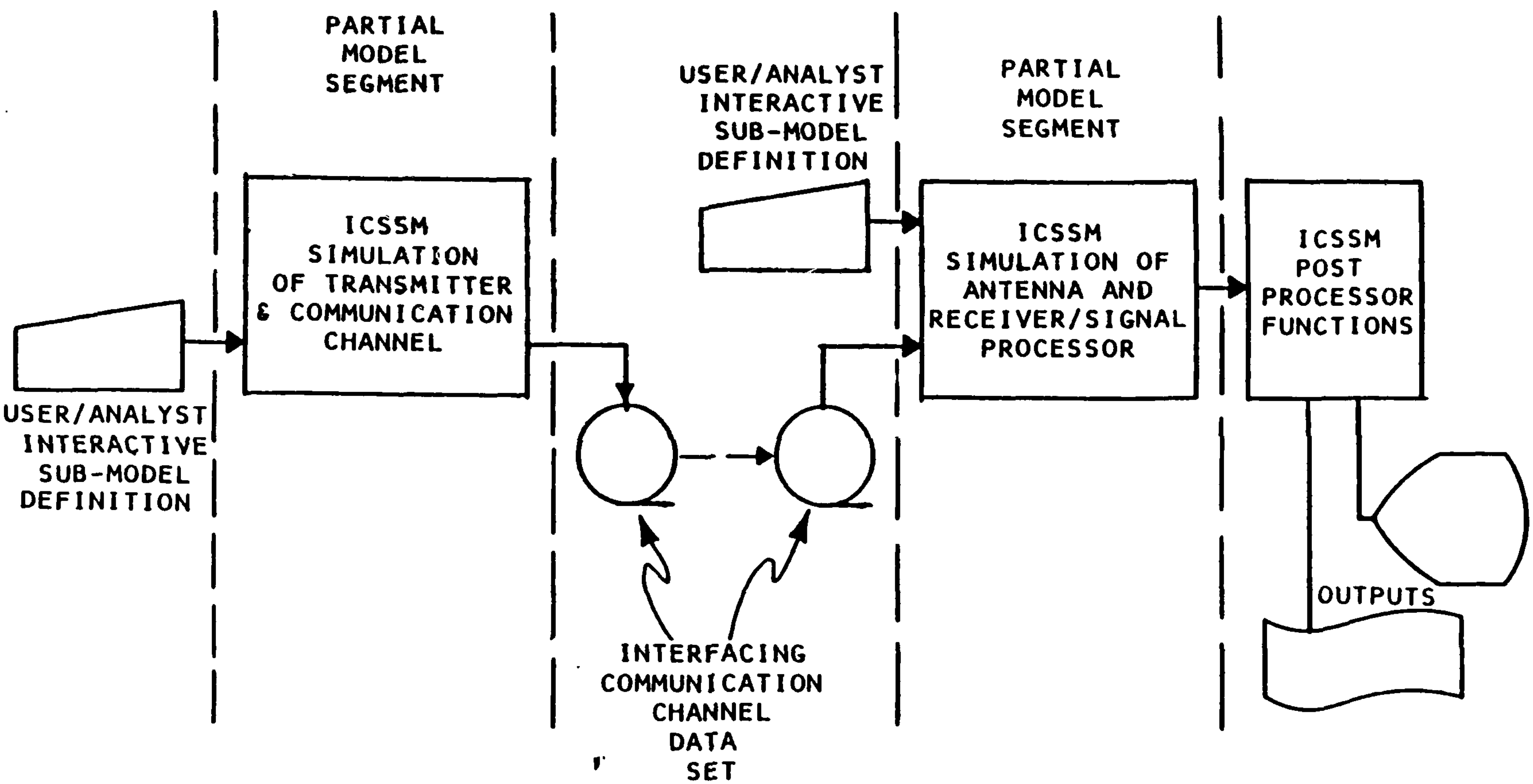


Figure 2-6. ICSSM Usage - An Example of the Partial Models Method

conditions of transmission. Similarly, experimental data, taken on an actual channel, could be substituted for the simulated channel data.

2.6 ICSSM LIBRARY MODULE RE-ENTRANT DESIGN

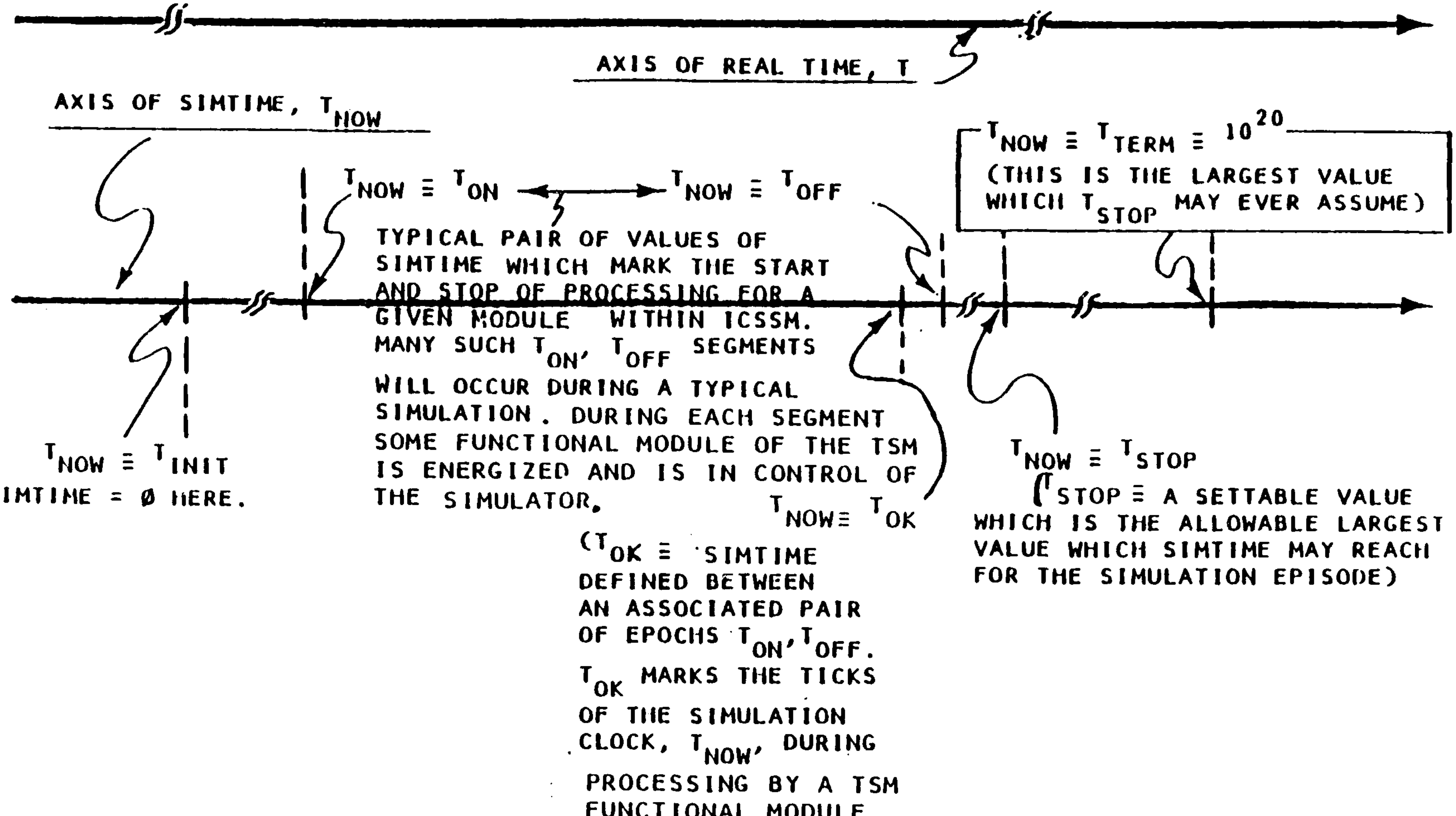
The ICSSM TSM incorporates specific functional modules that may be selected by the user in the model configuration step. By this means, the general structure of the simulation executive is rendered specific, thereby subsuming the simulation requirements of the particular TSM. Communications-theoretic processing and algorithmic manipulations are performed in "attached" applications modules. The applications modules are re-entrant; data needed as input by a particular module is made available to it through a collection of FORTRAN COMMON areas. Control signal and parameter values required by the modules are similarly provided. Operations on the data are performed by the modules without altering contents of the COMMON blocks (input areas). Results of the module operations are placed in FORTRAN COMMON blocks (output areas) for subsequent use by other elements of the TSM. When all operations of the module have been terminated, all particular data resulting from operations of that module reside in main storage areas - the module internal structure remains unaltered.

Individual module operation is initiated within the TSM when particularly designated conditions have been established in the control sections of the COMMON area. These conditions are the same for all applications modules. Initiating module operation is coincident with the T_{ON} time-marks of figure 2-7. Termination of module operation is coincident with the pair-wise corresponding T_{OFF}. During module execution, SIMTIME is clocked by the segment-time variable T_{OK}.

By means of comparisons among specific values of T_{ON}, T_{OK}, and T_{OFF} (say, T_{ONi}, T_{OKi}, T_{OFFi} compared with T_{ONi+1}, T_{OFFi+1}), concurrent processes manifested in the originating communications system model are converted into sequential processes occurring within the TSM.

2.7 EVENT STEP SIMULATION

The ICSSM is an event-driven or event-step simulator. Event-step programming is a useful technique for developing efficient and fast-running computer models. The advantages over pure time-step programs are: (1) events can be scheduled at time intervals appropriate to the processes they represent; thus, when little activity is occurring in a given module, the events can be widely spaced in time, while for very



GENERAL TIME RELATIONS AMONG SIMTIME VALUES

(A) $T_{INIT} \leq T_{NOW} \leq T_{STOP} \leq T_{TERM}$

(B) $T_{ON_I} \leq T_{OK_I} \leq T_{OFF_I}$

(C) $T_{INIT} \leq T_{ON_1} \leq T_{OK_1} \leq T_{OFF_1} \cdots\cdots \leq T_{ON_2} \leq T_{OK_2} \leq T_{OFF_2} \leq T_{ON_3} \leq T_{OK_3} \leq T_{OFF_3} \cdots \leq T_{STOP}$

Figure 2-7. SIMTIME (Simulation Time) Relationships within the ICSSM

active modules, the events can be closely spaced; (2) events can be scheduled on a real-time basis for some occurrences and on a "zero-time" basis when they represent pseudo-events (such as the actions of a bit-error computation module); (3) the Event List can be user-defined and new event types can be added as required; (4) events are controllable in that each process can define an event-time for strobing the system and can schedule events for activating other interconnected system elements.

ICSSM models actually use both event-step and time-step simulation management:

a. ICSSM employs a simulation clock that measures simulated real-time (SIMTIME) as it transpires during exercises. The diagram of figure 2-7 describes the time relationships that exist during any simulation episode.

b. ICSSM uses an Event Queue, the entries of which either are relatable to explicit processes occurring, or cause explicit processes to occur, during simulation. The diagram of figure 2-8 shows the hierarchical relationships that are established among SIMTIME, events in the Event Queue, and the "nodes" of the communications system being represented in the TSM.

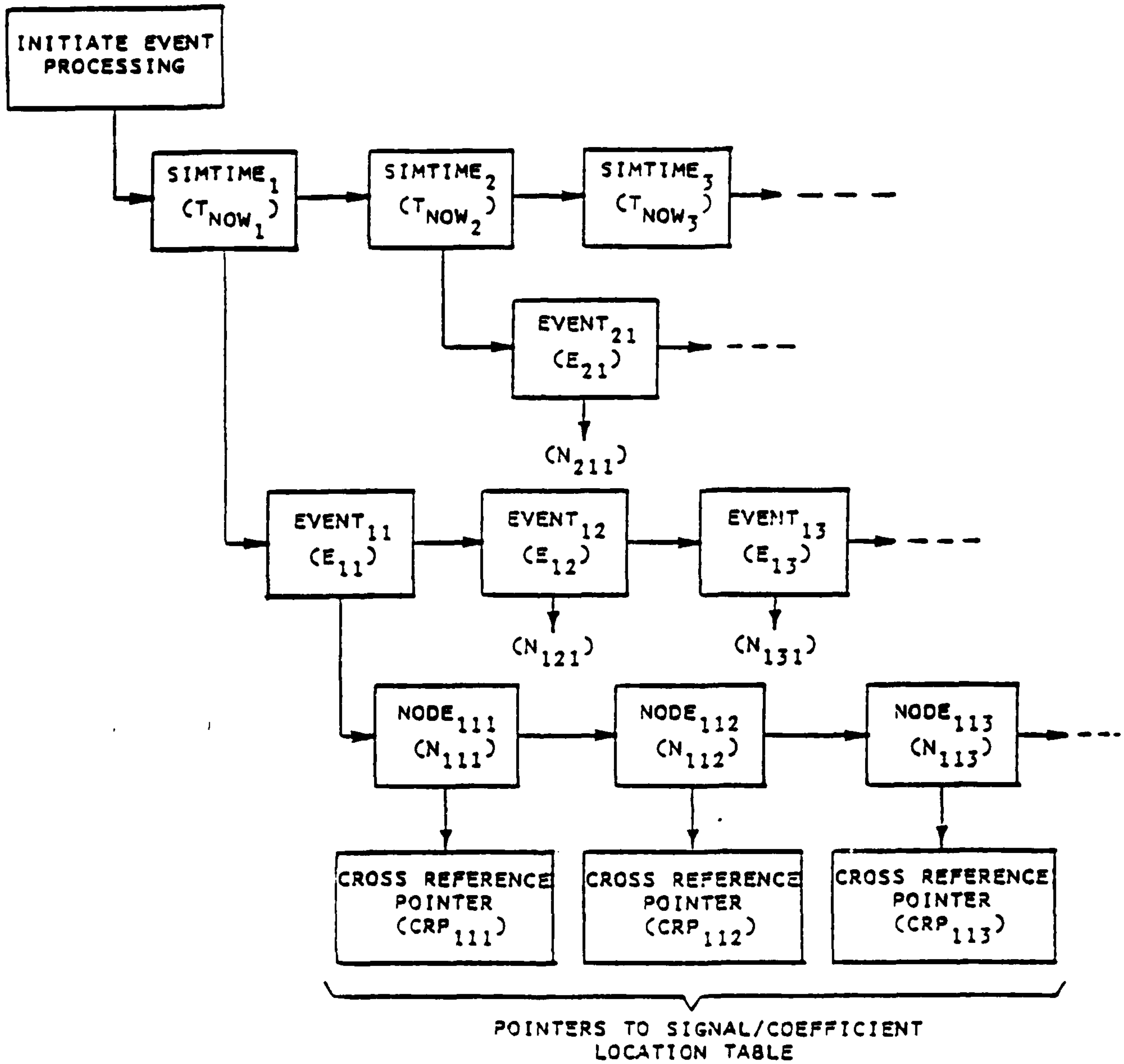


Figure 2-8. Event Queue Table - Hierarchy of Entries

SECTION III

IMPLEMENTATION OF THE ICSSM SYSTEM

3.1 INTRODUCTION

This section describes the implementation of the ICSSM system. The description is in three parts:

- o Paragraph 3.2 et seq provides a general overview of the ICSSM system implementation.
- o Paragraph 3.3 et seq provides a more detailed discussion of the ICSSM Simulator Component implementation down to the computer program level.
- o Paragraph 3.4 et seq provides a more detailed discussion of the ICSSM Applications Library Component implementation down to the computer program level.

For more detailed explanation of ICSSM implementation, please refer to the various ICSSM documents (references 50-55).

3.2 GENERAL DESCRIPTION OF ICSSM SYSTEM IMPLEMENTATION

The ICSSM system-subsystem structure is described in figure 3-1. ICSSM is comprised of two components: a Simulation Component (SC), and an Applications Library Component (ALC).

The SC provides: (1) input and model formulation capability to a prospective user via an interactive interface; (2) control a. d housekeeping facilities needed to carry out simulation; and (3) output data reduction/display facilities for recording or examining simulation results.

The ALC contains algorithmic implementations of communication system functional elements and communications system test equipment or test/measurement methods. These may (through agencies of the SC) be incorporated into communications systems model formulated by a use be exercised together, and the results recorded.

The diagram of figure 3-2 indicates the rel.tionship between the ICSSM system and the user. The ICSSM system oversees the configuration of a simulation model representing a communications system under study. The ICSSM system also oversees the exercise of the model and the subsequent data reduction and display of results produced by the simulation.

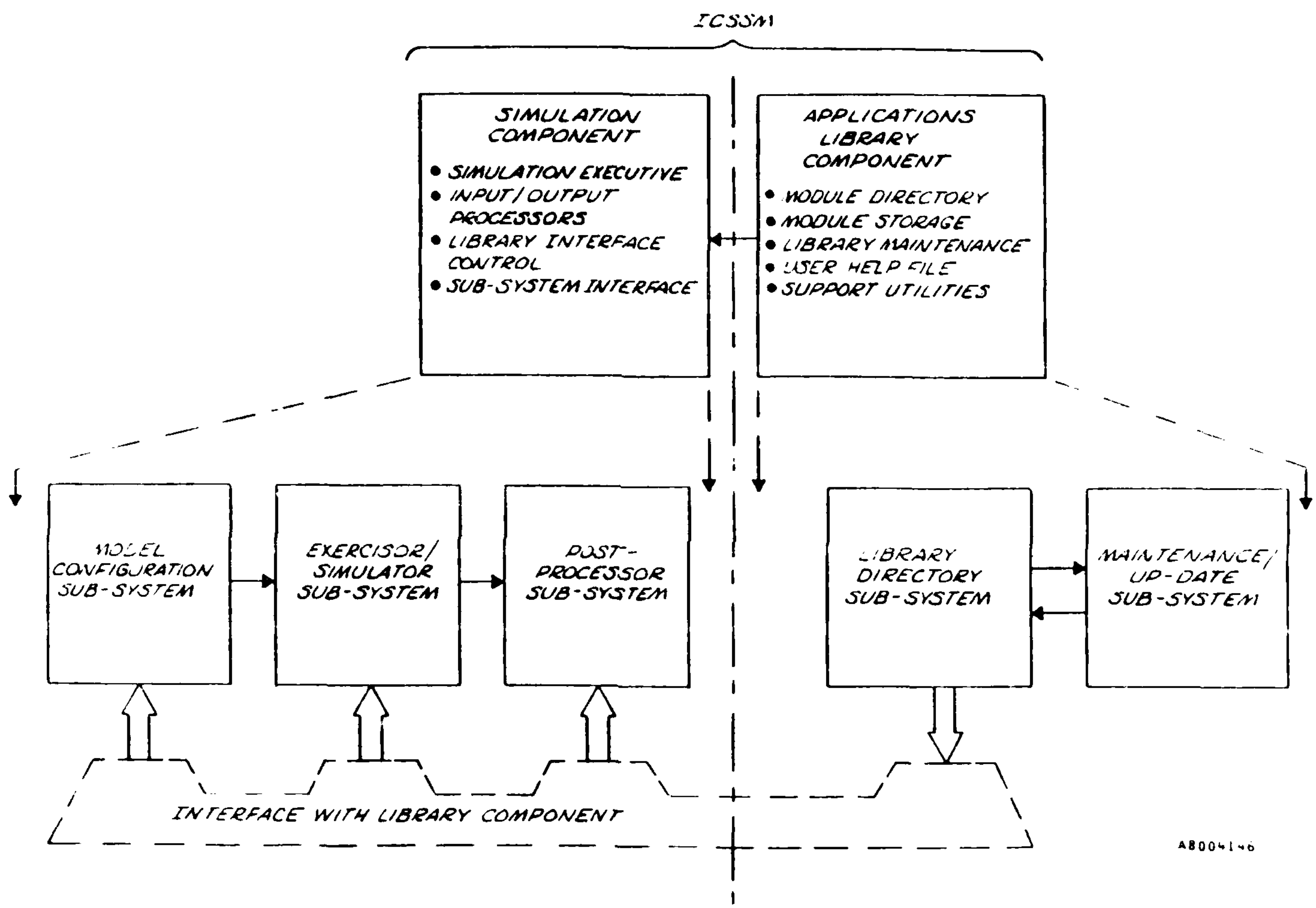


Figure 3 1. System and Subsystem Structure of ICSSM

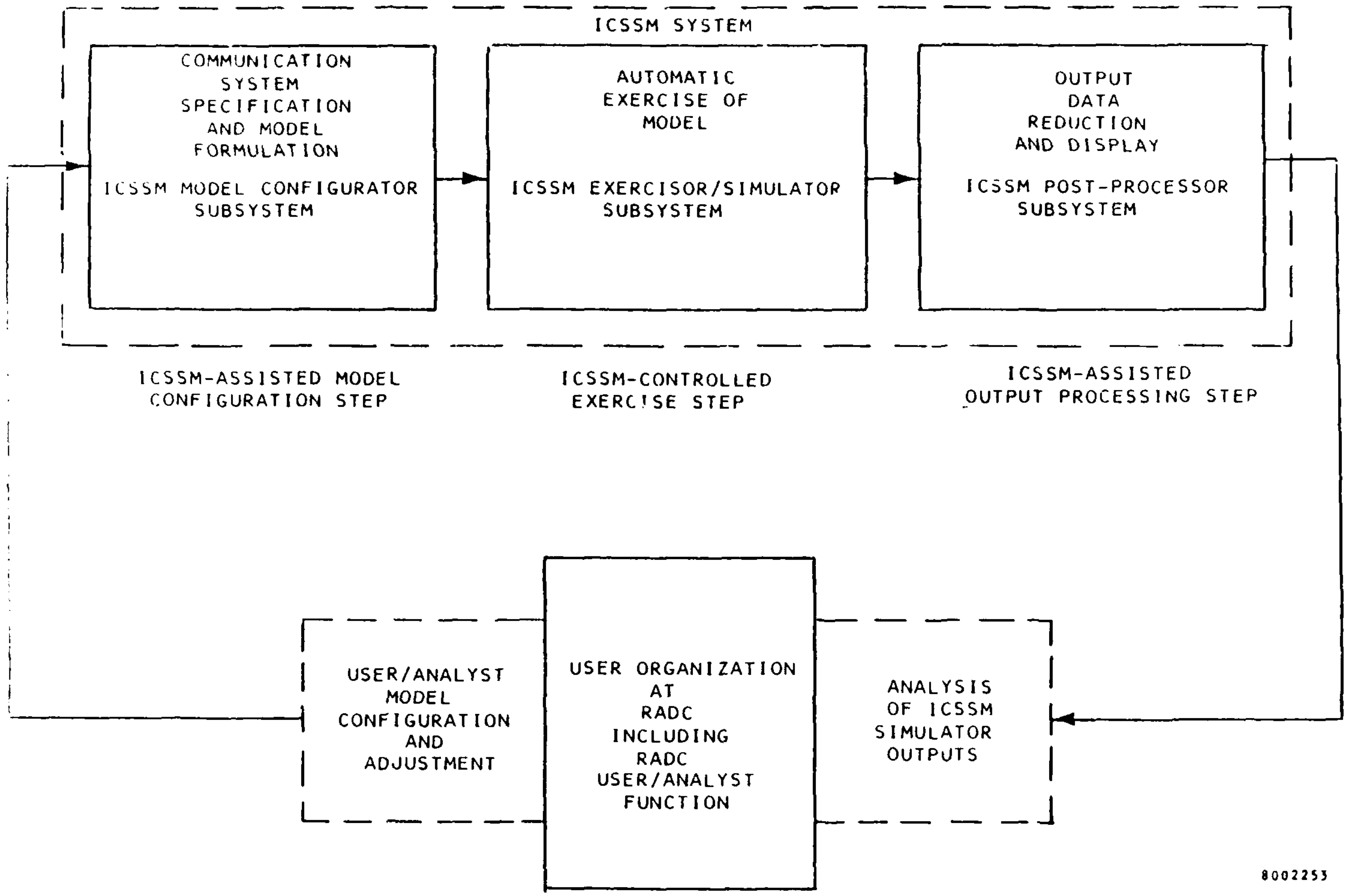


Figure 3-2. Relationships Between ICSSM and User

The subsequent paragraphs of the present section will describe the structure for each of the aforementioned components of the ICSSM.

3.2.1 General Organization of the ICSSM Simulation Component

The Simulation Component of ICSSM is comprised of three subsystems (see figure 3-1):

a. The Model Configurator (MC) subsystem.

b. The Exercisor/Simulator (ES) subsystem.

c. The Post-Processor (PP) subsystem.

The MC subsystem consists of two programs: the Select (MCS) program and the Precompiler (MCP) program; and associated computer files.

The MC subsystem provides facilities whereby a user interactively configures a communication system model. The configured-model information is used to produce a simulation model of the communications system of interest. The MC subsystem provides an interactive facility whereby the user, via a crt terminal, selects and specifies: (1) functional elements to be included in the model; (2) interconnections among the selected elements; (3) locations (ports) within the specified model from which output signals are to be drawn for subsequent data reduction and evaluation; and (4) requirements for checkpoint triggering.

The ES subsystem consists of the Exercisor Kernel (EK), and executive computer program, conjoined with a software version of the simulation model configured via the MC subsystem. The conjoint program so produced implements the communication system model ("target" simulation model or "TSM") of interest. The TSM may then be exercised under control of the host computer's operating system, as would any other application program, producing simulation results (data) that are stored on certain computer files for subsequent analysis.

The PP subsystem consists of two computer programs: the PP Selector (PPS) program, and the PP Exercisor (PPE) program. The data from the simulation executed by the ES subsystem may be submitted to the PP subsystem along with output port information contained in a computer file automatically generated by the MC subsystem. This information, together with additional data introduced via interactive facilities

provided by the PPS program, furnish the means whereby: (1) data reduction/data manipulation processes (eg, Fourier Transformation, Statistics) may be applied to the data resulting from the simulation episode supported in the ES subsystem and (2) results of these processes may be displayed to the user for his evaluation and subsequent use.

3.2.2 General Organization of the ICSSM Applications Library Component

The Applications Library Component depicted in figure 2-1 is comprised of two subsystems (see figure 3-1):

a. A Library/Directory (LD) subsystem.

b. A Maintenance/Update (MU) subsystem.

The LD subsystem is a group of computer files. Some of these files contain subroutines that are algorithmic embodiments of communications system functional processes (eg, matched filter, envelope detector, decoder, propagation channels, antennae), or are utility and support subroutines designed and intended for use in testing or validating ICSSM operation. Other files of the LD subsystem comprise a directory/index of the functional processes (modules) provided in the library files. The directory and library fields collectively supply the algorithmic support for the Simulation Component (SC).

The MU subsystem consists of a computer program that may be used to add functional modeling elements to the files of the LD system.

3.2.3 Subsystem Structures of ICSSM Components

This paragraph expands on the brief descriptions of the subsystems provided above. Both components of ICSSM are considered, and the relationships among the subsystems are described. The general flow of data and control within ICSSM is depicted in figure 3-3.

3.2.3.1 General Description of the ICSSM Simulation Component

Figure 3-4 shows that the MC subsystem consists of the Select (MCS) program, the Precompiler (MCP) program, and certain related files. Figure 3-5 shows that the ES subsystem consists of the EK program and the Process Module (PM).

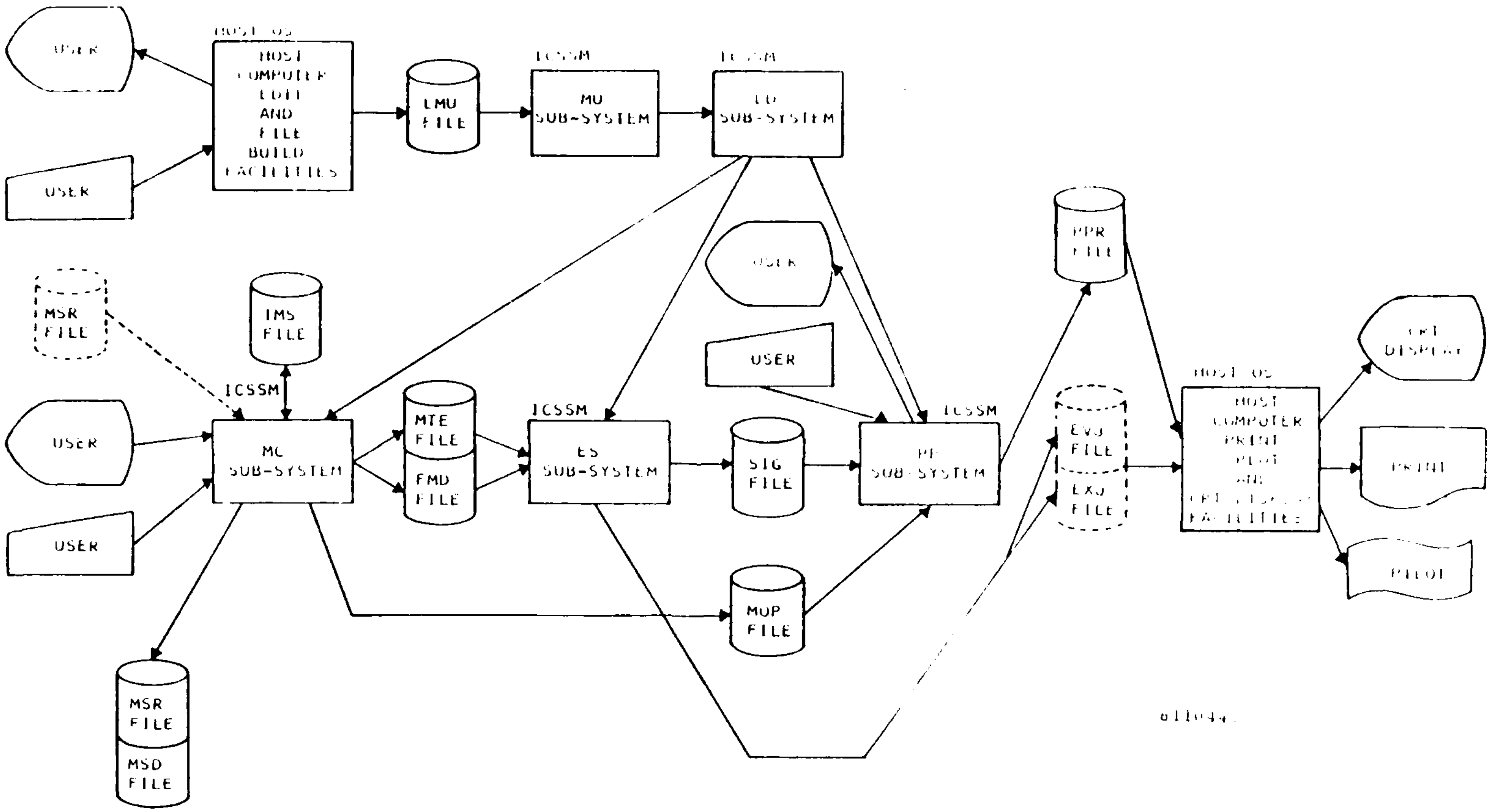


Figure 3-3. ICSSM System - General Flow

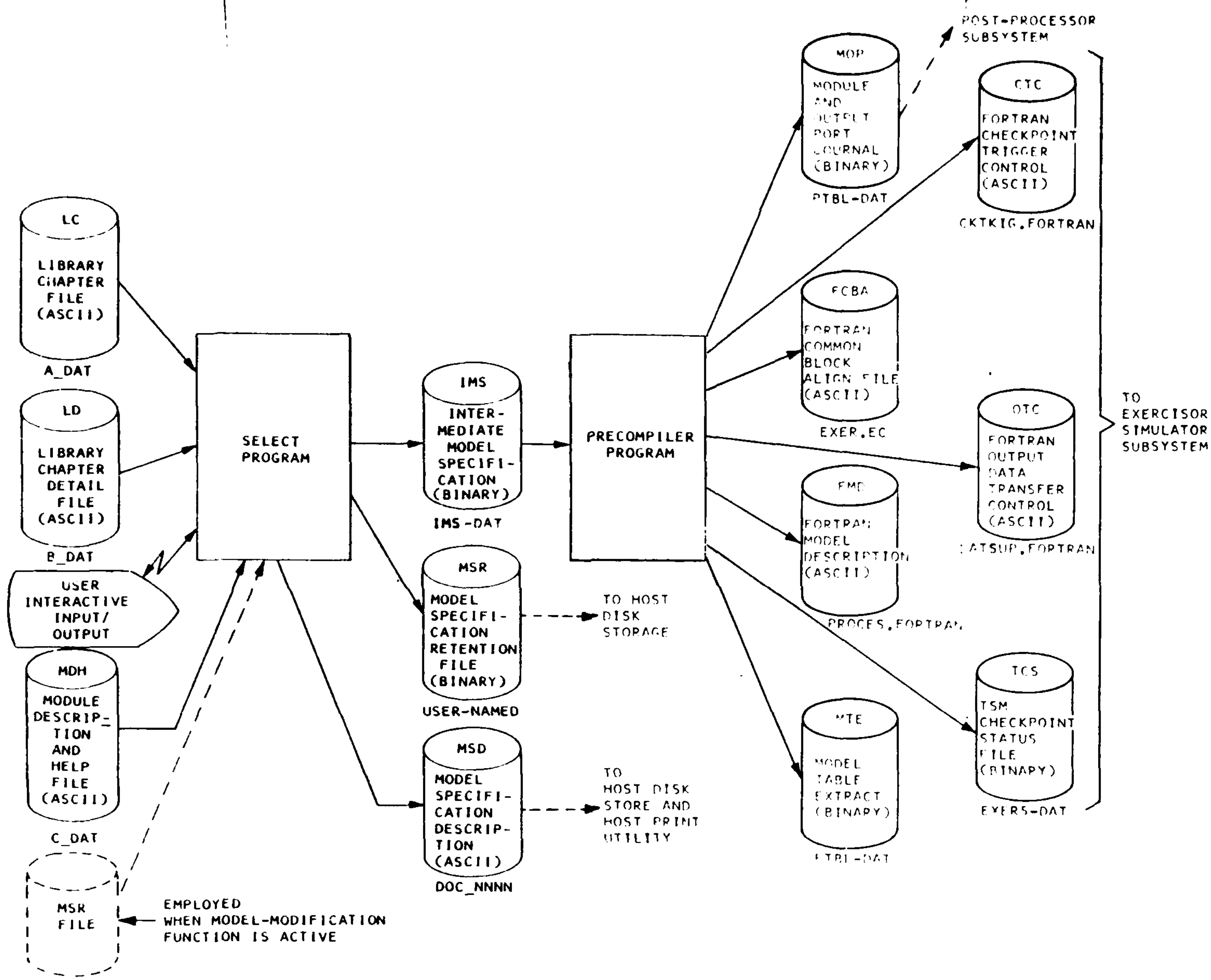


Figure 3-4. Structure of Model Configurator Subsystem (MCS)

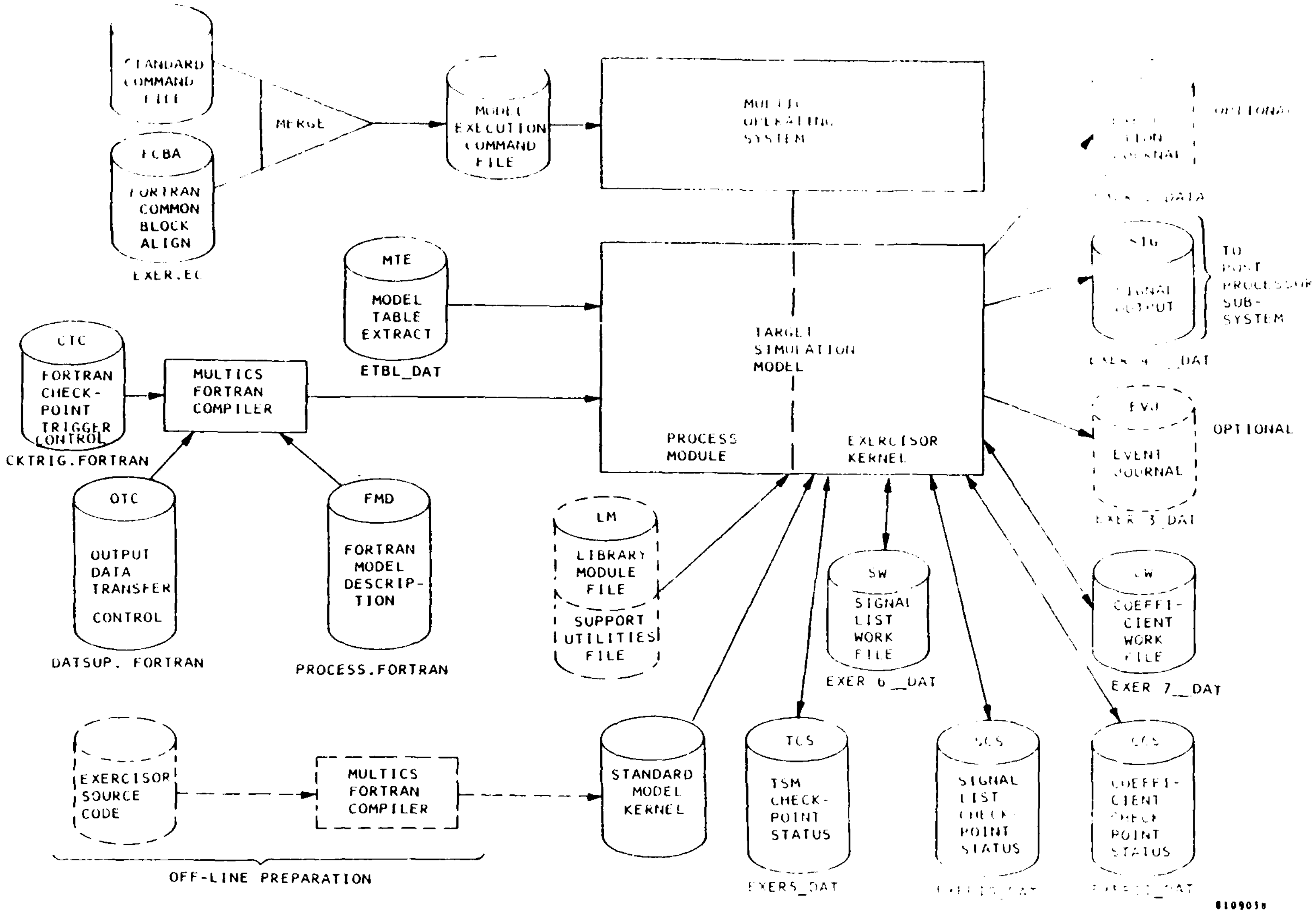


Figure 3-5. Structure of Exercisor/Simulator (ES) Subsystem

The PM is a specially prepared, model-dependent and automatically written FORTRAN subroutine that contains information describing the communication system model that was entered by the user in interactive operation of the MC subsystem. The PP subsystem consists of the Selector (PPS) program, the Exercisor (PPE) program, and certain related files as shown in figure 3-6.

3.2.3.1.1 General Description of the Model Configurator Select (MCS) Program. The MCS program operates interactively to:

- o Aid the user in reviewing the contents of the Library to locate suitable modeling elements for the simulation application of interest.

- o Prompt the user to provide required inter-element connections that reflect the simulation application of interest.

- o Provide means for accepting user-specified values for settable parameters that may be associated with modeling elements selected.

- o Provide means for accepting specifications for checkpoint control, for output data retention, and model modification.

The MCS program operates in two steps. The first step (or phase) of operation provides interactive facilities for:

- o Initial selection of Applications Library elements (viz, modules),

- o Registration of the values of settable parameters associated with the modules selected,

- o Specification of module interconnections,

- o Specification of checkpoint triggers, and

- o Specification of module output ports to be used for data retention.

The second phase provides interactive facilities for modification of:

- o Module selections,

- o Parameter values,

- o Interconnections,

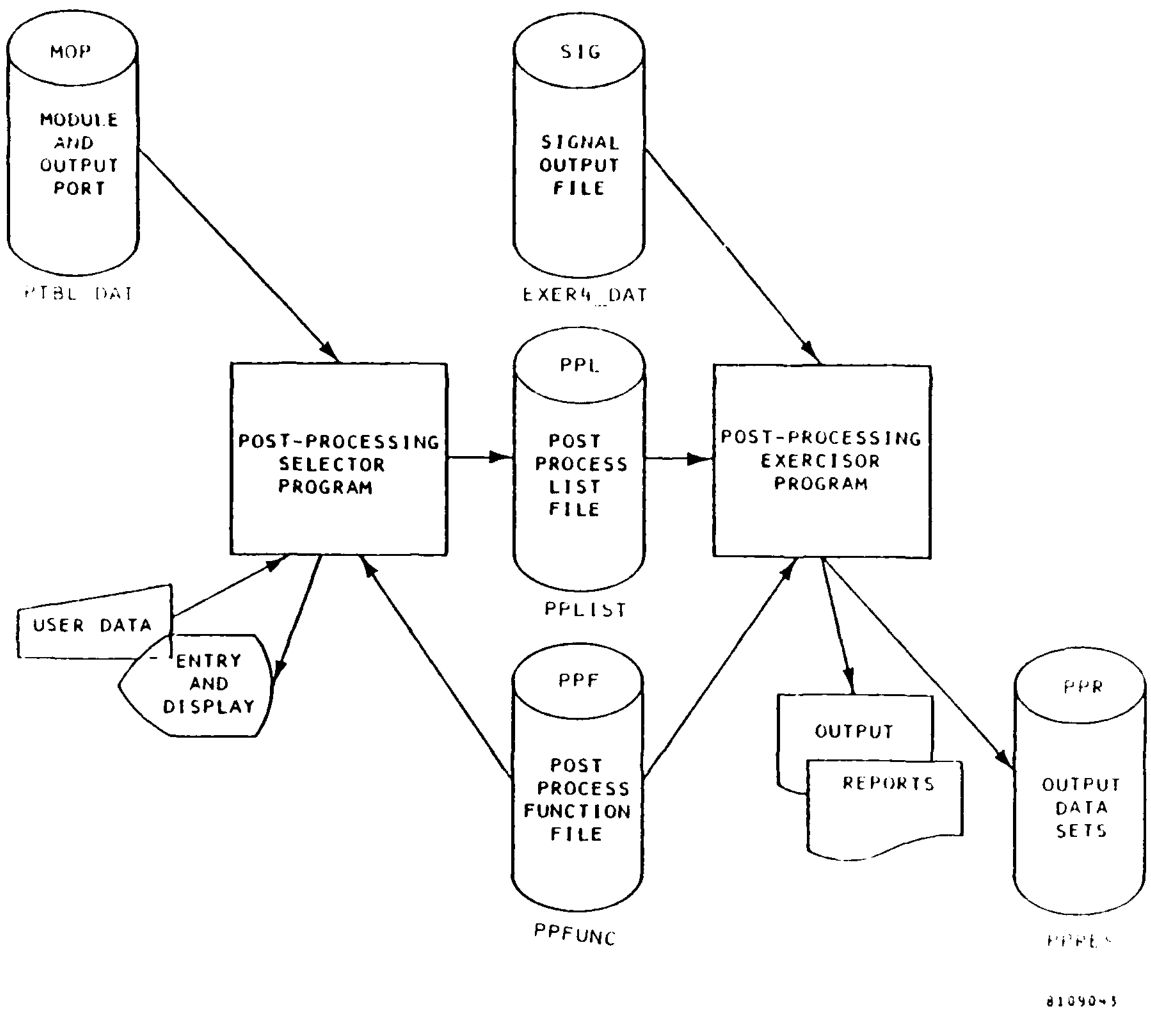


Figure 3-6. Structure of Post-Processor (PP) Subsystem

o Checkpoint triggers, and

o Output port selections.

Following initiation, the MCS program displays a "menu" on the user's crt terminal. The menu provides nine choices for use of the program. Termination of any subsequent interactive operation in the first phase of program operation returns the user to this menu display/select step or provides the user with a menu of alternative steps to direct subsequent actions of the program.

Selection and interconnection of modeling elements in the MCS program results in the generation of three data sets: the Intermediate Model Specification (IMS) file, the Model Specification Retention (MSR) file, and the Model Specification Description (MSD) file.

The IMS file contains a compacted description of the user-specified model and contains all the module identifiers, parameter values, interconnection information, and simulation-condition information needed to define the simulation to be performed. The IMS file is the vehicle of coupling to the Precompiler program and contains all the data needed to produce the FORTRAN Model Description (FMD) file, the FORTRAN Checkpoint Trigger Controller (CTC) file, and the FORTRAN Output Data Transfer Controller (OTC) file. These files are output products of Precompiler execution.

The Model Specification Retention (MSR) file contains compacted configuration tables, all module identifiers, parameter values, interconnection information, simulation-condition information, and configuration status information needed for future retrieval of the model specification by the MCS program as a basis for a modified configuration. The user is prompted by the MCS program to supply a file name for the MSR.

The Model Specification Description (MSD) file contains a table summarizing the modules employed in the model, the parameters (and assigned values thereof) associated with each module, the checkpoint trigger selections (and assigned triggering values thereof) associated with each module, the applicable output connections, and the specified output data transfer assignments for each module. The name of the MSD file is derived from the user-assigned file name for the MSR file by adding the prefix "doc_" to the file name specified for the MSR file.

The MCS program performs four kinds of validation of data input by the user. These validations implement consistency and completeness checks of the user model:

- o For each parameter associated with a module, there is an associated pair of values defining boundaries of a "reasonable value" range for the parameter. The user is prompted to assign a value for each parameter. If a value assignment is not within the given boundaries for that parameter, the MCS program detects the out-of-range condition, issues an appropriate diagnostic to the crt terminal, and directs the parameter value specification until a value within the prescribed boundaries is assigned.

- o At completion of model configuration, the MCS program checks the model for certain types of modules whose presence is necessary for model reasonableness. Each model must contain a "self-updating" or source module (to drive the simulation), at least one intermediate module (necesary for a "sensible" simulation model), and a terminating module. The MCS program detects the absence of any of these module types and issues the appropriate diagnostic to the crt terminal. Operation automatically returns to the main selection menu to allow the user to modify the model if a "self-updating" or a terminating module is not present. If an intermediate module is absent, the MCS program issues a warning and allows the user to determine whether to return to the main selection menu for model modification or to continue pre-output checking.

- o For each module used in the user model, the MCS program requires the specification of connections between each and every available model output port and <u>some</u> input port of <u>some</u> module. The MCS program also requires specification of a connection from <u>some</u> module to each and every input port of each module. The MCS program records in tables the connection data specified for each module used in the user model. At the completion of interconnection specification, and at the completion of configuration selection, the MCS program scans the nascent connections table looking for unconnected ports. For each unconnected port detected, an appropriate diagnostic is issued to the crt terminal, and the user is required to modify connection data to expunge the open-port condition that has been detected. The

user is permitted, at this point in model configuration, to add a new module to the model.

For each reassertion of output connections, the completeness of port connections is re-checked, as specified above, and the nascent connections table scanned again in search of open ports, as described above. The MCS program continues the interactive validation sequence specified until connection conditions being validated are fully satisfied.

- At completion of model configuration, the MCS program determines if at least one port has been selected as an output source of simulation-processed data. If no ports have been assigned to transfer data to the simulation output files, the MCS program detects this condition and issues the appropriate diagnostic to the crt terminal. The MCS program provides a default setting of ports for output data transfers or returns to the main selection menu to allow output-data transfer-port assignments. The user is directed to choose between the two aforementioned options.

At the successful completion of the validation process, IMS, MSR, and MSD files are written, thus ensuring completely validated model information for further processing. MCS program operation then terminates.

The IMS file is identified to the ICSSM host computer operating system with the name: ims_dat. The MSR file is identified to the ICSSM host computer operating system with a user-assigned name.

The MSD file is identified to the ICSSM host computer operating system by a name derived by adding the prefix "doc_" to the user-assigned name of the MSR file.

Input data for the MCS program is also obtained from the Library Chapter file, the Library Chapter Detail file, and the Module Description and Help file.

3.2.3.1.2 General Description of Model Configurator Precompiler (MCP) Program. The MCP program accepts the contents of the IMS file as input and produces output data in seven files (see figure 3-4):

- FORTRAN Model Description (FMD) file

- FORTRAN Checkpoint Trigger Controller (CTC) file

- FORTRAN Output Data Transfer Controller (OTC) file
- Model Table Extract (MTE) file
- FORTRAN Common Block Alignment (FCBA) file
- Module and Output Port (MOP) journal file
- TSM Checkpoint Status (TCS) file.

The IMS file is read sequentially, and information destined for the seven output files is extracted from the records thus read through the internal processing of the MCP program.

The FMD file contains a model-dependent version of the Process Module (ie, subroutine PROCES). The Process Module is the basic application-specific modeling module and, in conjunction with the Exercisor Kernel, makes up an individual, model-dependent "target simulation model" (TSM). That is, each communication system model submitted to ICSSM for simulation gives rise to a specific version of the PROCESS/EXERCISOR combination that constitutes the target simulator program. The FMD file contains an automatically generated FORTRAN version of the Process Module, which will be submitted subsequently to the ICSSM host computer FORTRAN compiler (see figure 3-5) for inclusion in the TSM.

The CTC file contains a model-dependent version of the Checkpoint Trigger Controller (ie, subroutine CKTRIG). The Checkpoint Trigger Controller is referenced by both the Exercisor Kernel and the modules in the TSM to determine when checkpointing of the TSM will be scheduled and performed. The CTC file contains an automatically generated FORTRAN version of the Checkpoint Trigger Controller (reflecting user checkpoint parameter assignments), which is submitted subsequently to the ICSSM host computer FORTRAN compiler (see figure 3-5) for inclusion in the TSM.

The OTC file contains a model-dependent version of the Output Data Transfer Controller (ie, subroutine DATSUP). The Output Data Transfer Controller is referenced by the Exercisor Kernel for the regulation of simulation output data transfer to the simulation output files. The OTC file contains an automatically generated FORTRAN version of the Output Transfer Controller (reflecting user assignment of open output ports for data transfers), which is submitted subsequently to the ICSSM host computer FORTRAN compiler (see figure 3-5) for inclusion in the TSM.

The FCBA file contains a model-dependent set of SFC commands. This file is generated in compliance with the requirement (when executing FORTRAN programs under MULTICS operating system) to perform

in-main-storage alignment of FORTRAN common blocks among a FORTRAN mainline and its CALLED subroutines.

The TCS file contains an initializing-simulation time value generated by the MCP program. The Exercisor Kernel references the TCS file to determine if the TSM is to be initiated or re-initiated from some checkpointed status.

The MTE file contains tables, derived from the IMS file contents, reflecting modeling-element/module-interconnection data for the communications model being simulated (viz, the TSM) and control data used to regulate and direct the TSM during the course of execution. The tables consist of:

- o Master Module Table - a list of modules selected for the simulation along with their pertinent parameter and control data
- o Node List - a list of connection nodes that exist in the model being simulated
- o Parameter Lists - lists of sets of parameter values associated with each of the modules comprising the model
- o To-List - a list of module interconnection data that fixes and records the model topology
- o Checkpoint Triggering Values List - a list of triggering values for checkpoint parameters in the model.

The MOP file contains a table of those output ports selected for output data transfer to file for the model being simulated. These output ports are identified with Library Name, user name, and port number. The MOP file is an input to the Post-Processor subsystem.

The FMD, CTC, OTC, TCS, and MTE files provide data input to the Exercisor/Simulator subsystem (see figure 3-5). They are identified to the ICSSM host computer operating system as the PROCES.FORTRAN file, the CKTRIG.FORTRAN file, the DATSUP.FORTRAN file, and EXER5_DAT file, and the ETBL_DAT file, respectively.

3.2.3.1.3 <u>General Description of the Exercisor/Simulator (ES) Subsystem/Target Simulation Model (TSM) Program</u>. Each TSM program consists of a fixed or programmatically constant simulation executive (Exercisor Kernel) and a customized simulation model segment (Process Module) automatically constructed by prior operation of the MCS and MCP programs. These are programatically combined to form a complete customized simulator, known as the TSM. Each modeling conception rendered to the

ICSSM system, via the interactive MCS program, gives rise to a unique version of the TSM. Exercise of the resultant TSM produces simulation output data reflecting the behavior of simulated communication signals appearing at each connection node of the originating communication system model. Output data is stored in the ICSSM system's Signal Output file for subsequent analysis. The TSM is the center of simulation activity in the ICSSM system.

Principle elements that comprise the Exercisor/Simulator (ES) subsystem are shown in figure 3-5, along with ancillary computer system elements. The FMD file produced by the MC subsystem is submitted to the ICSSM host computer's FORTRAN compiler. By this mechanism, the model-specific Process Module (PM) segment is created for use with the Exercisor Kernel (EK). The EK (FORTRAN-coded and submitted previously to the FORTRAN compiler) is available in an executable form, invariant from TSM to TSM.

3.2.3.1.4 General Description of the Post-Procesosr Selector (PPS) Program. The principle elements of the Post-Processor subsystem are depicted in figure 3-6.

The Post-Processor Selector (PPS) program provides interactive crt-terminal-oriented access to the ICSSM system to:

- Aid the user in selecting post-processing functions, algorithms, and operations
- Prompt the user to designate to which TSM signal output the post-processing functions are to be applied
- Accept user specified values for settable parameters associated with the post-processing function(s) selected.

In summary, the PPS program is used to specify the output data reduction and algorithmic processing to be applied to data generated by execution of an ICSSM TSM.

The PPS program interacts with the user, via a crt terminal, in a prompt/response mode to retrieve and display:

- Tutorial information describing the capabilities and use of the ICSSM post-processing facilities
- Information on algorithmic processes that might be applied to signals generated within the TSM

- Information concerning the origin and nature of the TSM output signals that are accessible for post-processing.

Based on user responses, the PPS program automatically generates all instructions and prescriptions needed in carrying out the data reduction and processing designated.

3.2.3.1.5 General Description of the Post-Processor Exercisor (PPE) Program. The Post-Processor Exercisor (PPE) program accepts instructions and execution parameters produced by PPS program execution and processes TSM-generated signals accordingly. The signals processed are available to it in the Signal Output (SIG) file. The results of PPE execution are available for subsequent display and examination.

3.2.3.2 General Description of the ICSSM Applications Library Component

The subsystem organization of figure 3-1 indicates that the Applications Library Component (ALC) is composed of two subsystems: the LD subsystem (figure 3-7) and the MU subsystem (figure 3-36).

The LD subsystem is comprised of six data sets organized into an Applications group and a Utilities group. These files are described in paragraphs 3.2.3.2.1 and 3.2.3.2.2.

The MU subsystem is comprised of a single program - the Library Maintenance/Update (LMU) program. This program is described in paragraph 3.4.2.

3.2.3.2.1 General Description of the Library/Directory Applications Group (LDAG). The LDAG provides on-line storage for the application modules (functional elements required for communications system modeling), which reside in the Library Module (LM) files. These files contain all subroutines that may be used in configuring the TSM.

The LM files are grouped conceptually into "Chapters," based upon the taxonomy used in classifying the modules contained in the LD. The Library Chapter (LC) file contains data on the most general classification of application modules.

The Library Chapter Detail (LD) file contains data describing those application modules assigned to the chapters delineated in the LC file. The structure of these two files

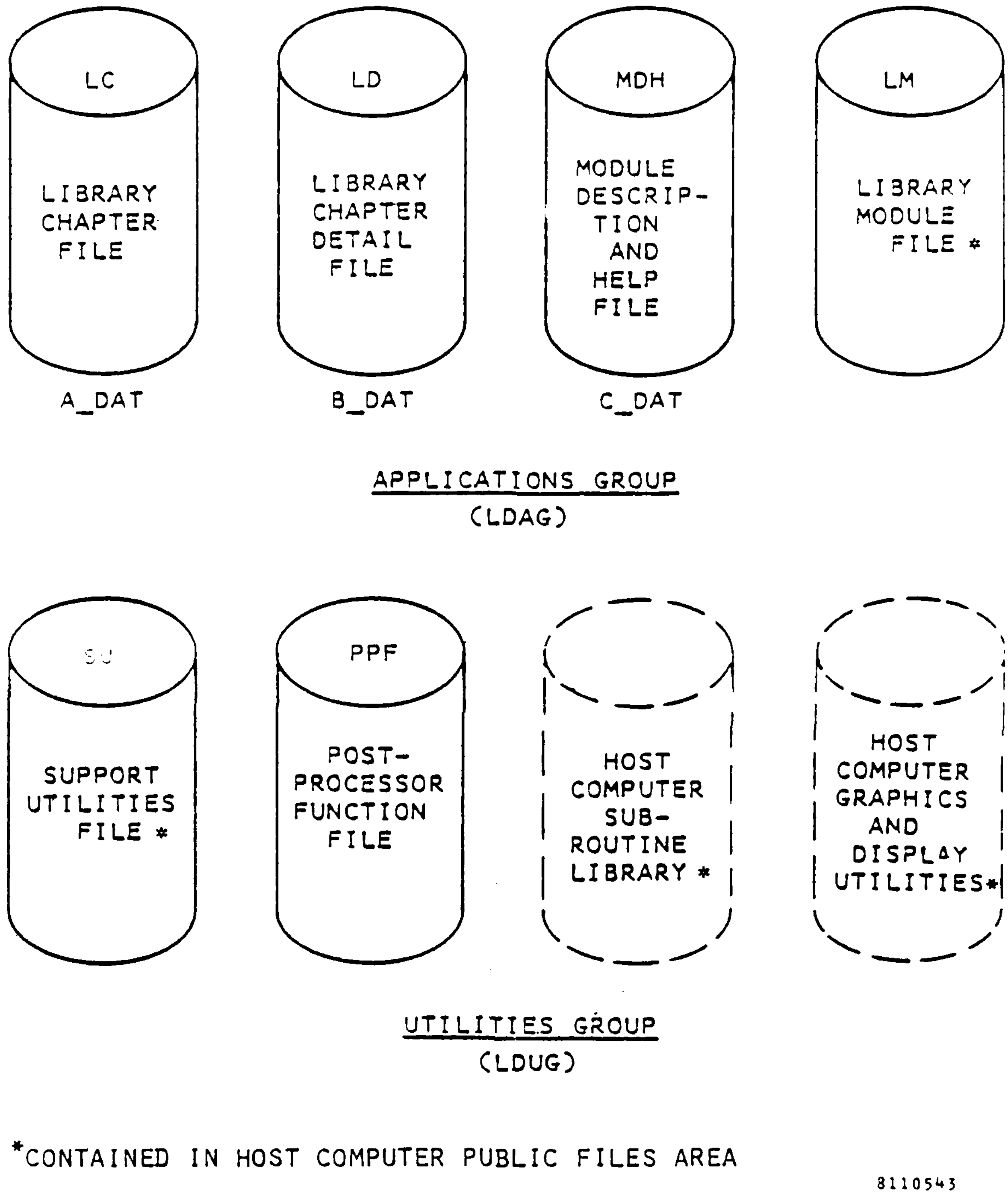


Figure 3-7. Structure of Library/Directory (LD) Subsystem

is analogous to chapter headings and within-chapter details provided by the "Table of Contents" of most textbooks.

The Module Description and Help (MDH) file contains an entry for each application module registered in the LM file. The user accesses the MDH file using the facilities of the MCS program.

The hierarchic structure of the LDAG is described in figure 3-8.

The LM files store FORTRAN subroutines used in constructing TSM. Two classes of subroutines exist in the LM files: Class 1 subroutines which are cataloged in the LD file are employable directly as model elements in configuring the TSM; and Class 2 subroutines which are not cataloged in the LD file are dependent subroutines employed by the subroutines in Class 1, but not directly as modeling elements.

Class 1 subroutines are referred to as "modules." Class 2 subroutines are referred to as "dependent modeling subroutines" (DMS).

3.2.3.2.2 General Description of the Library/Directory Utilities Group (LDUG). The LDUG stores subroutines and other program elements required by the programs that constitute the ICSSM system. The LDUG is comprised of the Support Utilities (SU) files and the Post-Processor Function (PPF) file.

SU files contain subroutines that may be used by programs of the ICSSM system as required (eg, internal to the ICSSM executive software or internal to the modules residing in the LM file). The SU files are not used directly as communications modeling elements but perform common data-manipulative or control functions and services required in the programs of the ICSSM system.

The PPF file entries are used automatically by the PPE program in data reduction and signal processing operations performed upon signals derived from the TSM, as designated in normal operation of the ICSSM facilities.

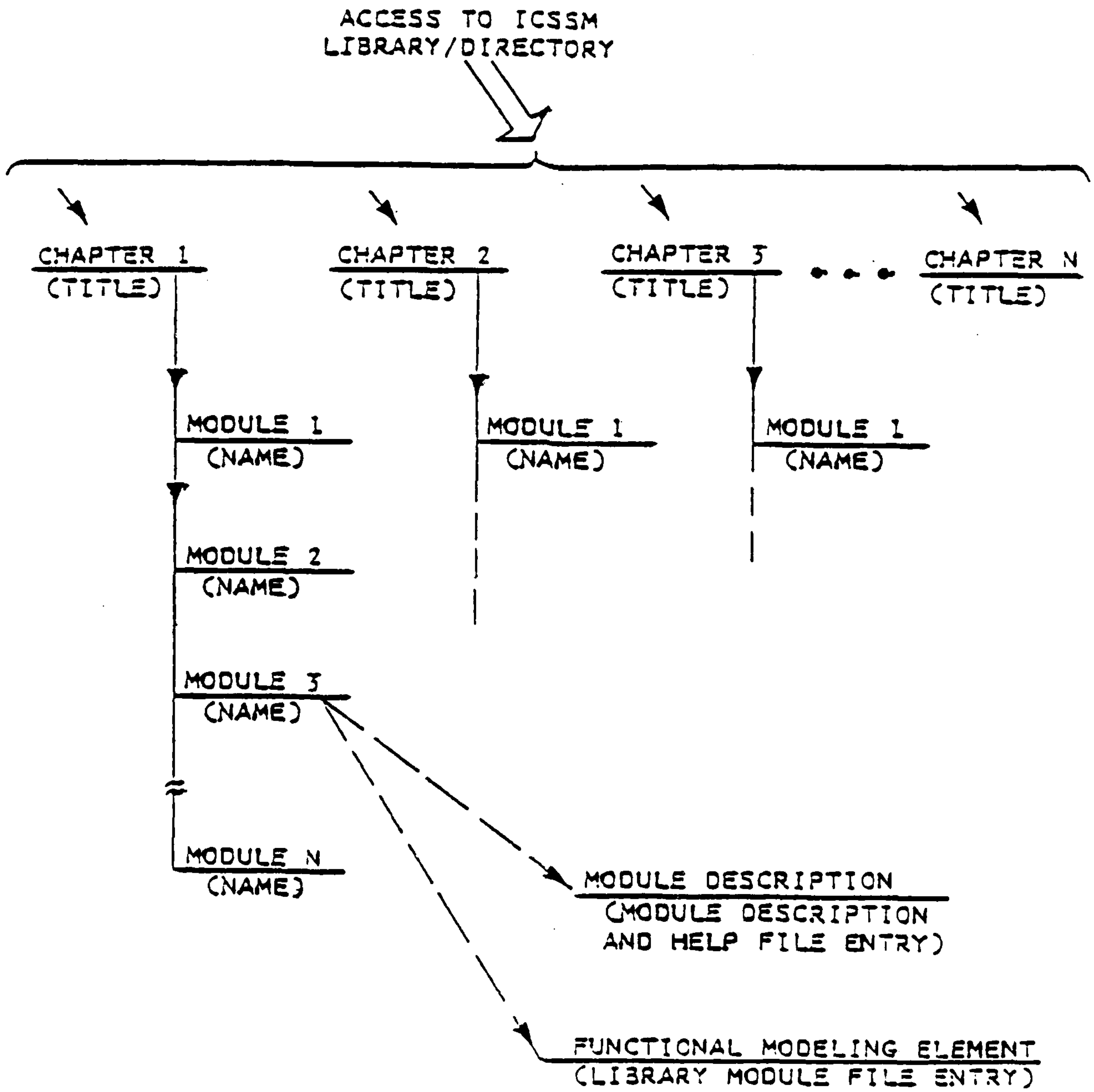


Figure 3-8. ICSSM Library Directory Applications Group
(LDAG) - Hierarchic Organization

NOTE

Figure 3-7 indicates two additional "files" within the LDUG. These are the host computer scientific subroutine library (SSL) and the host computer graphics and display utilities (GDU). While these "files" do not form a part of the ICSSM software per se, they are conceptually a part of the total processing capability employable by the ICSSM user.

3.3 DETAILS OF ICSSM SIMULATION COMPONENT IMPLEMENTATION

The following paragraphs describe the programs, subroutines, and files comprising the ICSSM Simulator Component. The programs described are:

- o Model Configurator Select (MCS)
- o Model Configurator Precompiler (MCP)
- o Exercisor/Simulator (the Kernel of each TSM)
- o Post-Processing Selector (PPS)
- o Post-Processing Exercisor (PPE)

3.3.1 Description of the MC Select (MCS) Program

The MCS program provides terminal-oriented access to the ICSSM system.

Execution of the selection and interconnection phases of the MCS program results in the generation of the Intermediate Model Specification (IMS) file.

The IMS file contains an abstracted, compacted description of the specified model and contains all the module identifiers, parameter values, checkpoint triggering requirements, output data retention requirements, and interconnection information needed to define the simulation to be peformed. The IMS file is the coupling vehicle to the MCP program. It contains all the data needed to produce the FORTRAN Model Description (FMD) file. The FMD file is the output product of the MCP program execution.

The MCS program uses as input files:

- o The Library Chapter (LC) file
- o The Library Chapter Detail (LD) file
- o The Module Description and Help (MDH) file.

The data from these files are used during the interactive TSM Module-selection/Model-configuration session afforded by the MCS program.

The MCS program implements model validation (ie, consistency and completeness) checks on user-input data (refer to paragraph 3.2.3.1.1. and table 3-1).

At the successful completion of the validation process, the IMS file is written, thus ensuring completely validated information for further processing. MCS program operation then terminates.

Program termination in the MCS program is automatic, requiring no overt action of the user, except in following the prompts presented to him.

3.3.1.1 Major Operations Performed in the MCS Program

The MCS program is organized according to the hierarchy of figure 3-9. The macro-level flowchart of figure 3-10 reflects the three phases in the MCS program operation. The first phase accepts selections of model specifications with

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 1 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
o Invalid selection from a menu. User inputs a number other than choices:	"Enter selection number:"
preliminary questions main menu selection Chapter selection module specification	
- Table consolidation command.	
- Next-mode selection following the addition of a module.	
- Modification selection for parameter modification.	
- Checkpoint trigger modification mode selection.	
- Exercisor/module selection for additional checkpoint triggers.	
o Absence of specifications. User requests a function that cannot be performed because of the absence of certain specifications in the current configuration:	
- Module deletion.	"There are no active modules in the present configuration. Strike any key to continue."
- Display of modules in current configuration.	

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 2 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
- Interconnection specification.	"There are no input ports in present configuration. Strike any key to coninue."
- Interconnection modification.	
- Display of input ports in current configuration.	
- Output data transfer specification.	"There are no output ports in present configuration. Strike any key to continue."
- Output data transfer modification.	
- Display of output ports in current configuration.	
- Parameter modification.	"There are no modules containing parameters in the present configuration. Strike any key to continue."
- Display of checkpoint triggers.	"There are no checkpoint triggers for the present configuration. Strike any key to continue."
- Checkpoint trigger specification edit.	"No checkpoint triggers have been specified."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 3 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
o Invalid selections from a specification list. User selects any number of specifications from a list of identifying numbers, of which at least one is incorrect.	
- Selection of checkpoint parameters.	"_______ selection(s) -- not valid."
- Selection parameters for parameter value modification.	
- Selection of checkpoint triggers for specification edit.	
- Selection of checkpoint triggers for deletion from checkpoint tables.	
- Addition of Exercisor checkpoint.	
o Checkpoint table overflow conditions. User requests too many parameters to be used as checkpoint triggers.	
- Selection of Exercisor checkpoint parameters.	"Only ______ additional checkpoint triggers may be specified at present."
- Selection of checkpoint triggers from module parameters.	"Use the first ______ parameters chosen as triggers? (y or n)."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 4 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
- Addition of Exercisor checkpoint parameters using the modify mode.	
- Addition of checkpoint triggers from module parameters using the modify mode.	
o Attempt to use a selection function when modify function applies. User requests to reuse a selection routine.	
- Selection of Exercisor checkpoint triggers.	"To specify further checkpoint triggers from the set of Exercisor parameters, use the Modify capability. Strike any key to continue."
- Interconnection Specification.	"For further interconnection specification, use the Modify capability. Strike any key to continue."
- Output Data Transfer Ports Specification.	"To change the status of any other output port with respect to post-processing requirements, use the Modify capability."
o Invalid selections for interconnection specification. User inputs selection numbers that correspond to deleted or nonexistent specifications.	
- Interconnection specification of connecting-output-port.	"Unavailable output port specified."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 5 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
- Interconnection modification of connecting-output-port.	"Input number associated with the connecting output port for ______ is invalid. Input port number, descriptor."
- Specification of input ports for interconnection modification.	"Unavailable input ports have been designated. Unavailable port specified: Unavailable input port number: ______. Strike any key to continue."
o Incomplete interconnection specifications. User has not specified connection data for all input and output ports.	
- Incomplete interconnection topology.	"Ports not connected. Port No. ______ Port descriptor ______ Output Port: ______ Input Port: ______ At present, there are more output ports than input ports in the configuration." "Interconnection specification cannot be complete when output ports outnumber input ports in a configuration."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 6 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
o Specification of out-of-range parameter value. User inputs a value for a parameter that is not within the assigned value range for that parameter.	
- Parameter value specification.	"Parameter out of range. Input new assigned value for parameter number, parameter descriptors."
- Parameter value modification.	"Parameter out of range. Input new assigned value for parameter number, parameter descriptors."
o Invalid specification of output port for output data transfers. User inputs an output port number that is non-existent or was deleted.	
- Specification of output ports for output data transfers.	"Specification of unavailable output port number: ______."
- Modification of output ports for output data transfers.	
o Incomplete model specifications detected by final model error-checking facilities.	
- Check model for active modules.	"There are no active modules in the present configuration. Strike any key to continue."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 7 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
- Interconnection status flags indicate unstarted or incomplete interconnection specifications.	"No interconnection specification. Choose interconnection specification from specification mode menu. Strike any key to continue."
	"Interconnection specification incomplete. Choose modify function from specification mode menu display. Strike any key to continue."
- Model does not contain the required essential modeling elements.	"Error: configured model does not contain essential self-updating driver module. Automatic return to specification mode menu. Strike any key to continue."
	"Error: configured model contains more than 1 self-updating driver module. Automatic return to specification mode menu. Strike any key to continue."
	"Error: configured model does not contain a terminating module. Automatic return to specification mode menu. Strike any key to continue."
	"Warning: configured model does not contain an intermediate module. Return to specification mode menu? (y or n)."

Table 3-1. Error Checking/Data Validation in the MCS Program (Sheet 8 of 8)

Program Segment/Activity Where Error Will Be Detected	Prompt and/or Error Messages
- Output data transfer requirements not specified prior to a change in configuration that may have affected these requirements.	"Warning: available output ports in configuration have been added and/or deleted since the beginning of configuration or since the assignment or ports open for post-processing. Return to specification mode menu? (y or n)."
- No ports specified for output data transfers.	"Error: no ports open for post-processing. Set default ports-open? (y or n) (if not, automatic return to specification mode menu. Select post-processing requirements)."

respect to the modules selected, the parameter values assigned, the checkpoint triggers selected and triggering values assigned, the interconnection data specified, and the output data transfer requirements assigned. This phase provides parameter value and interconnection topology validity checking.

The second phase accepts modifications of the model specifications comprising the model selected in the first phase. This phase also provides parameter value and interconnection topology validity checking when appropriate.

The third phase performs overall model completeness and consistency validation with respect to the: presence of essential types of modeling elements, interconnection topology, assignment of output data transfer requirements, and consistent modeling elements within a model. If a model cannot be validated with respect to the aforementioned checks, the third phase issues an appropriate diagnostic to

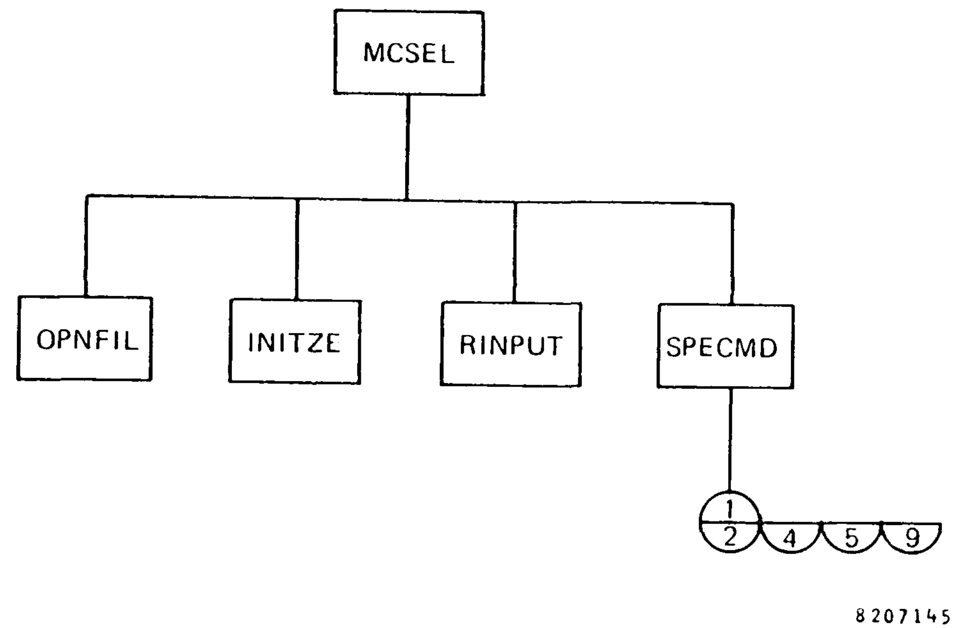


Figure 3-9. MCS Program Hierarchy (Sheet 1 of 9)

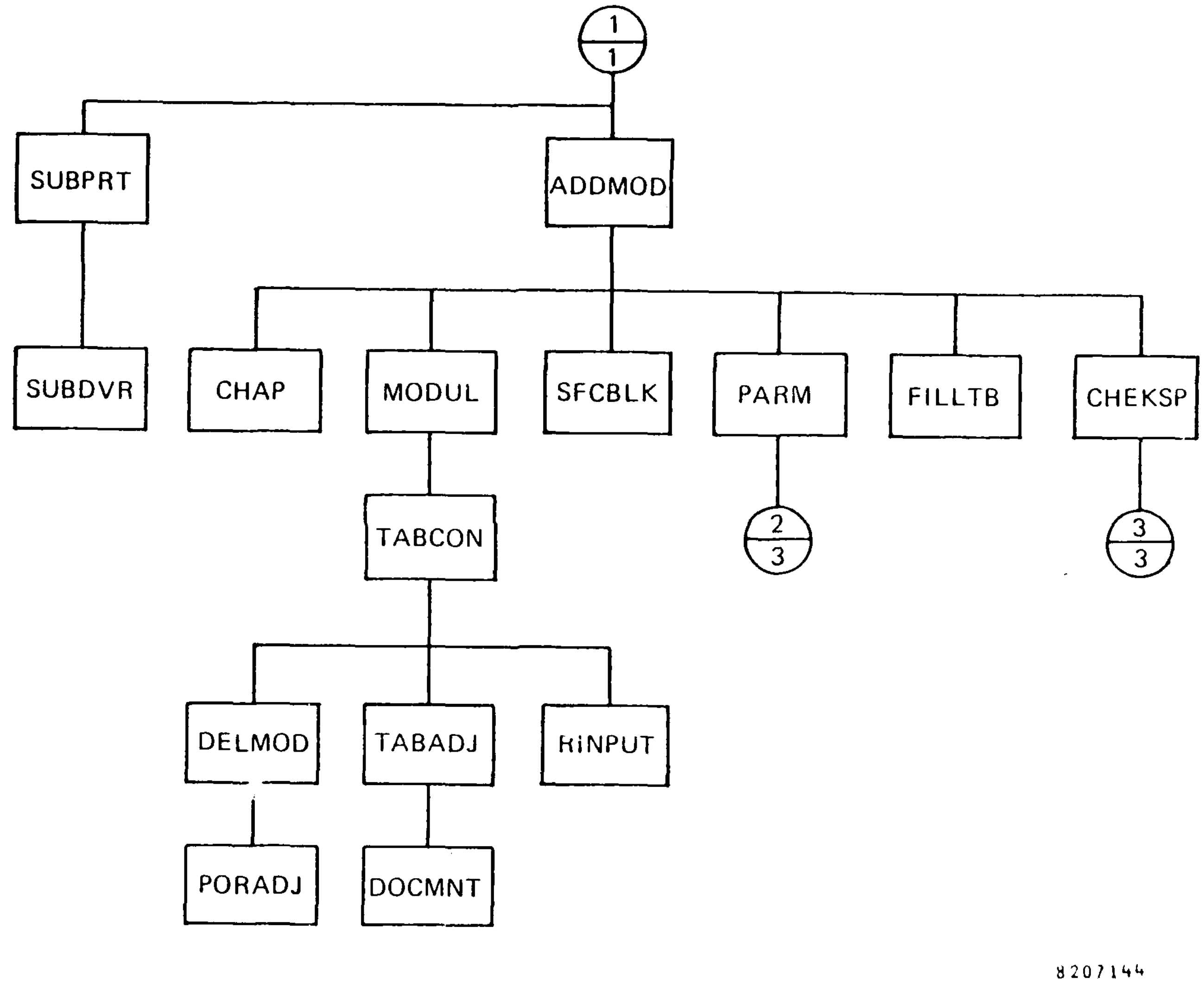


Figure 3-9. MCS Program Hierarchy (Sheet 2 of 9)

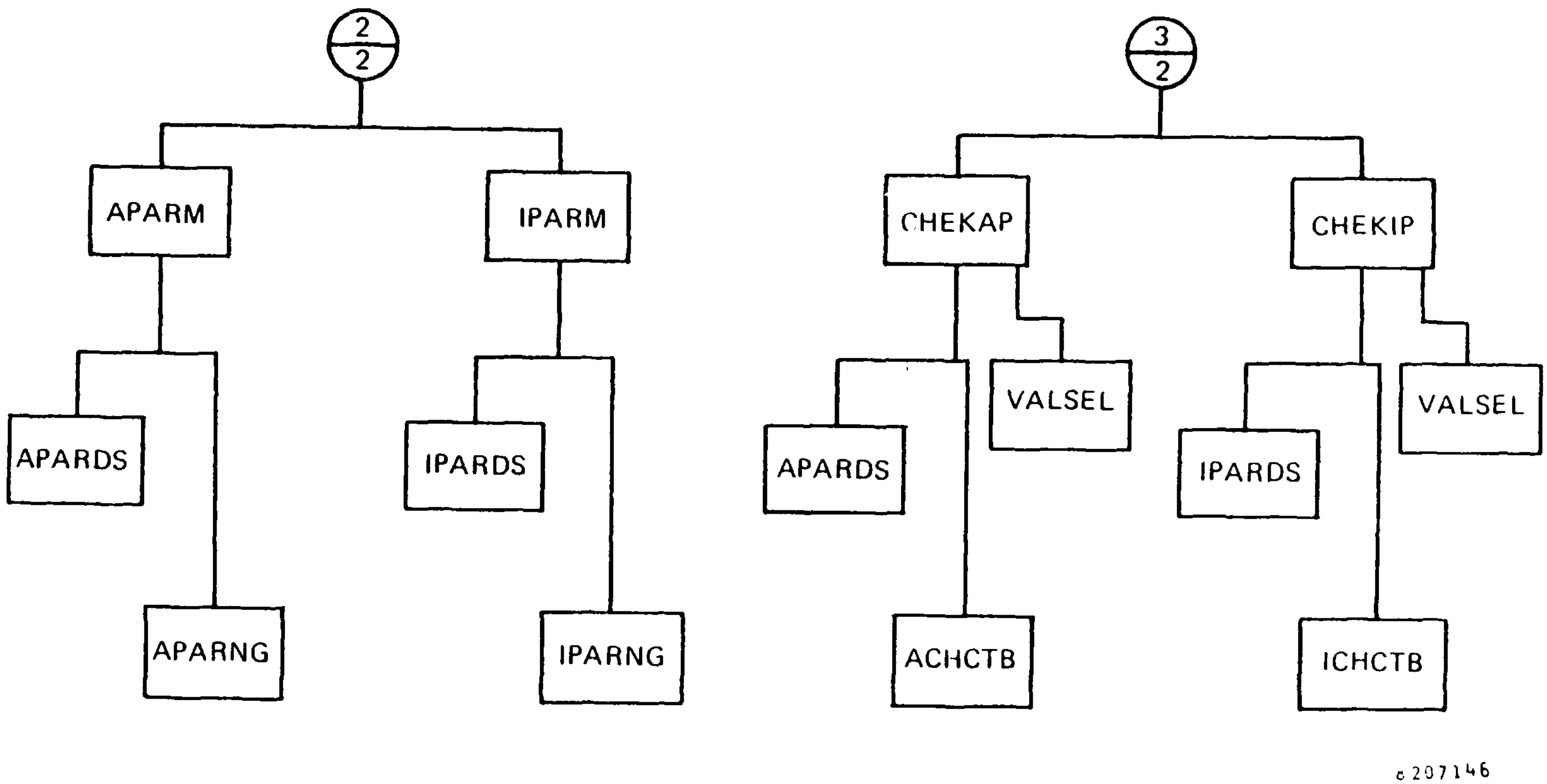


Figure 3-9. MCS Program Hierarchy (Sheet 3 of 9)

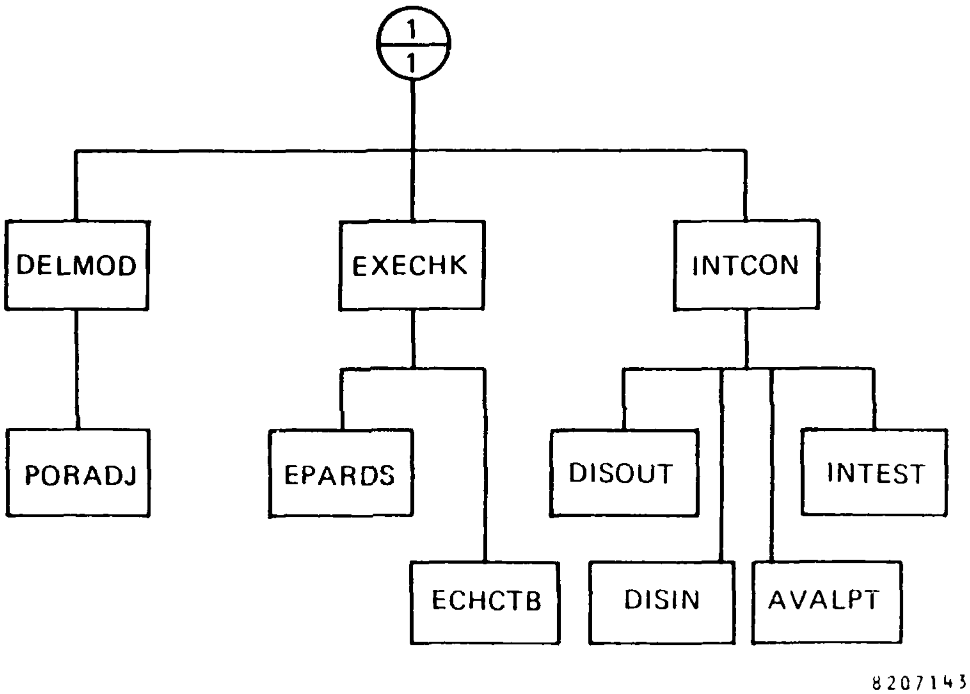


Figure 3-9. MCS Program Hierarchy (Sheet 4 of 9)

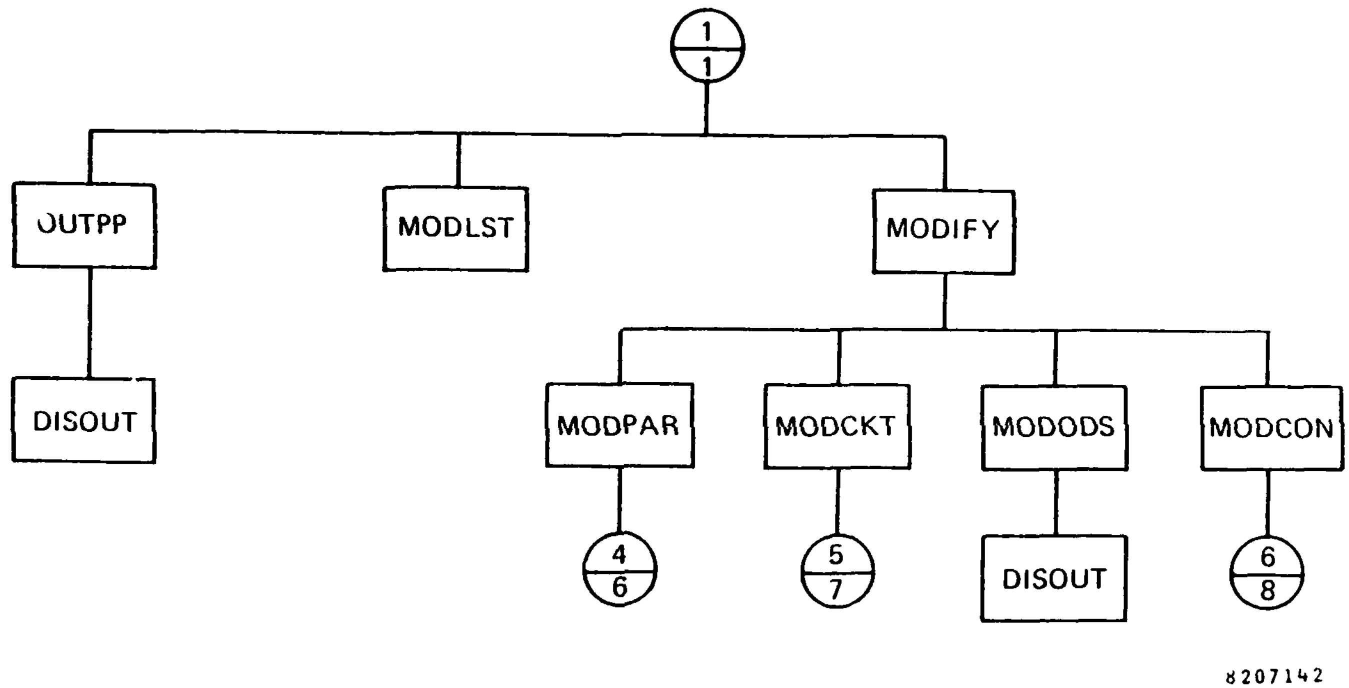


Figure 3-9. MCS Program Hierarchy (Sheet 5 of 9)

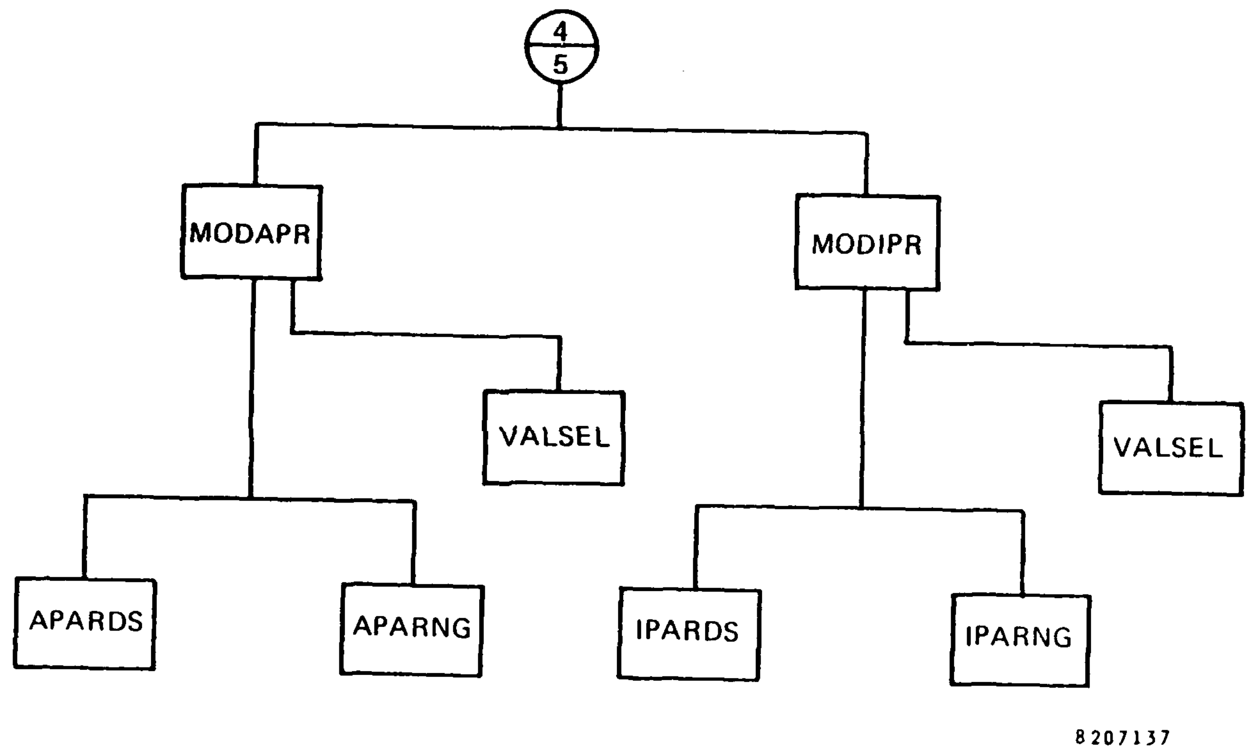


Figure 3-9. MCS Program Hierarchy (Sheet 6 of 9)

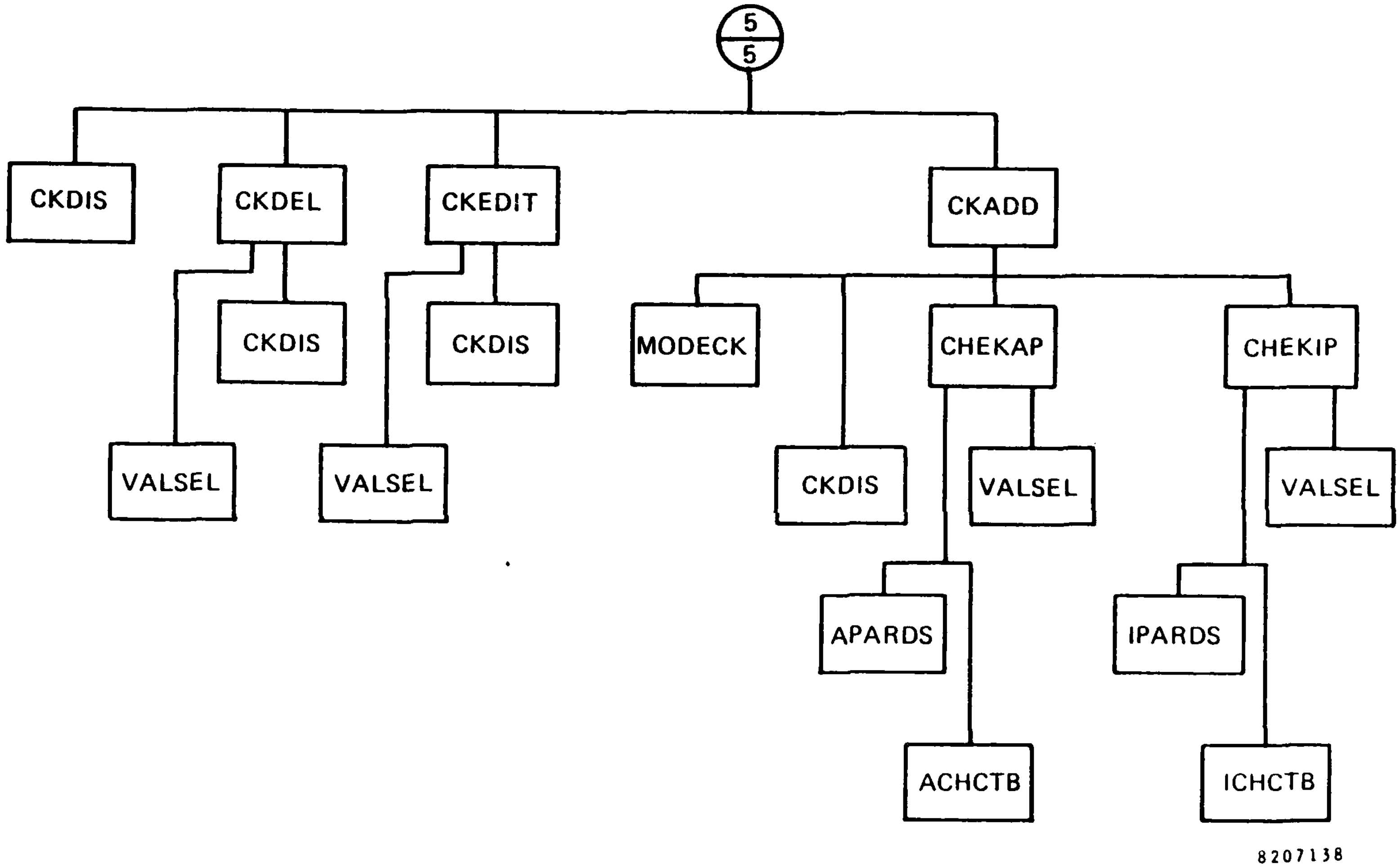


Figure 3-9. MCS Program Hierarchy (Sheet 7 of 9)

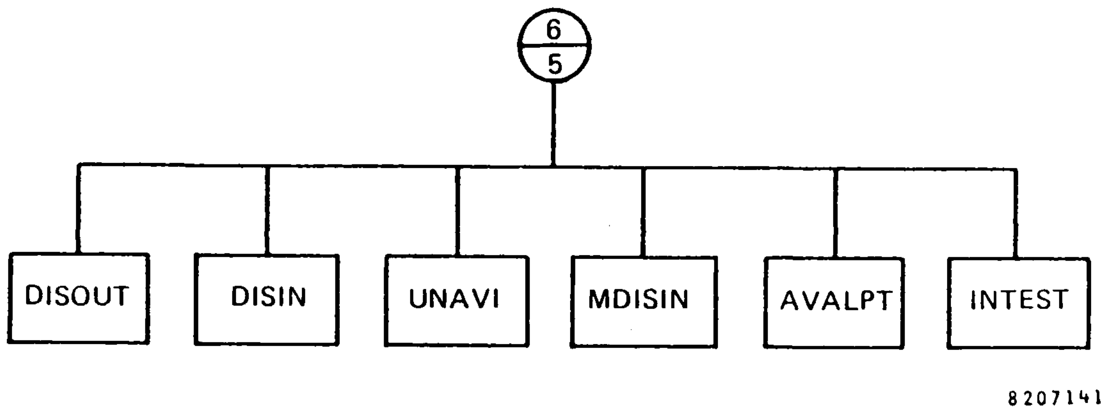


Figure 3-9. MCS Program Hierarchy (Sheet 8 of 9)

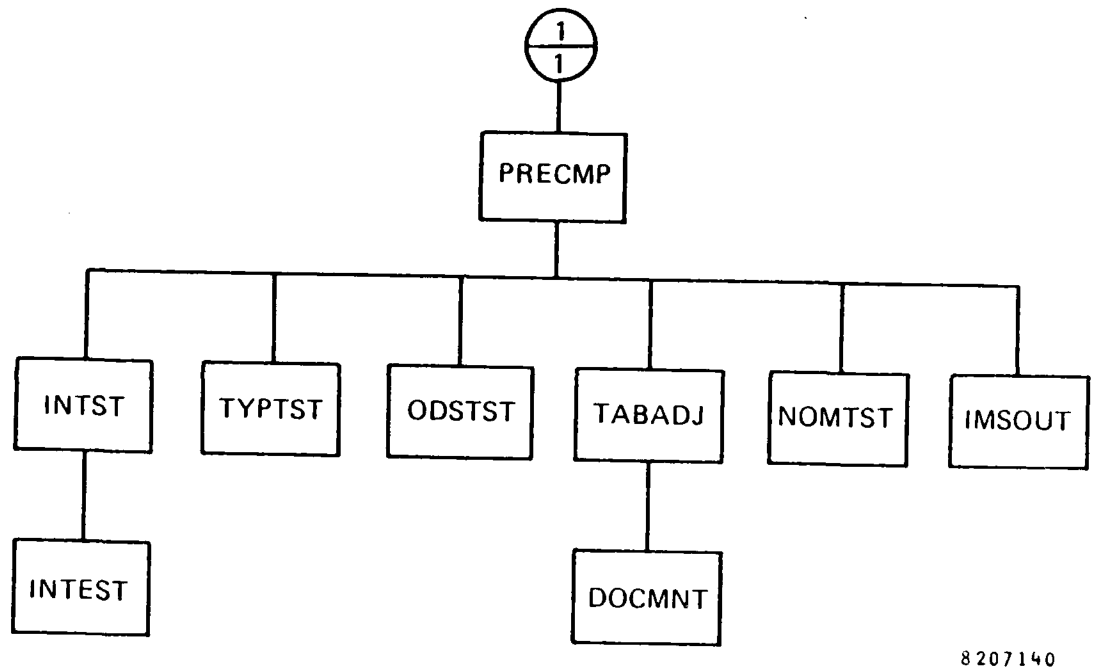


Figure 3-9. MCS Program Hierarchy (Sheet 9 of 9)

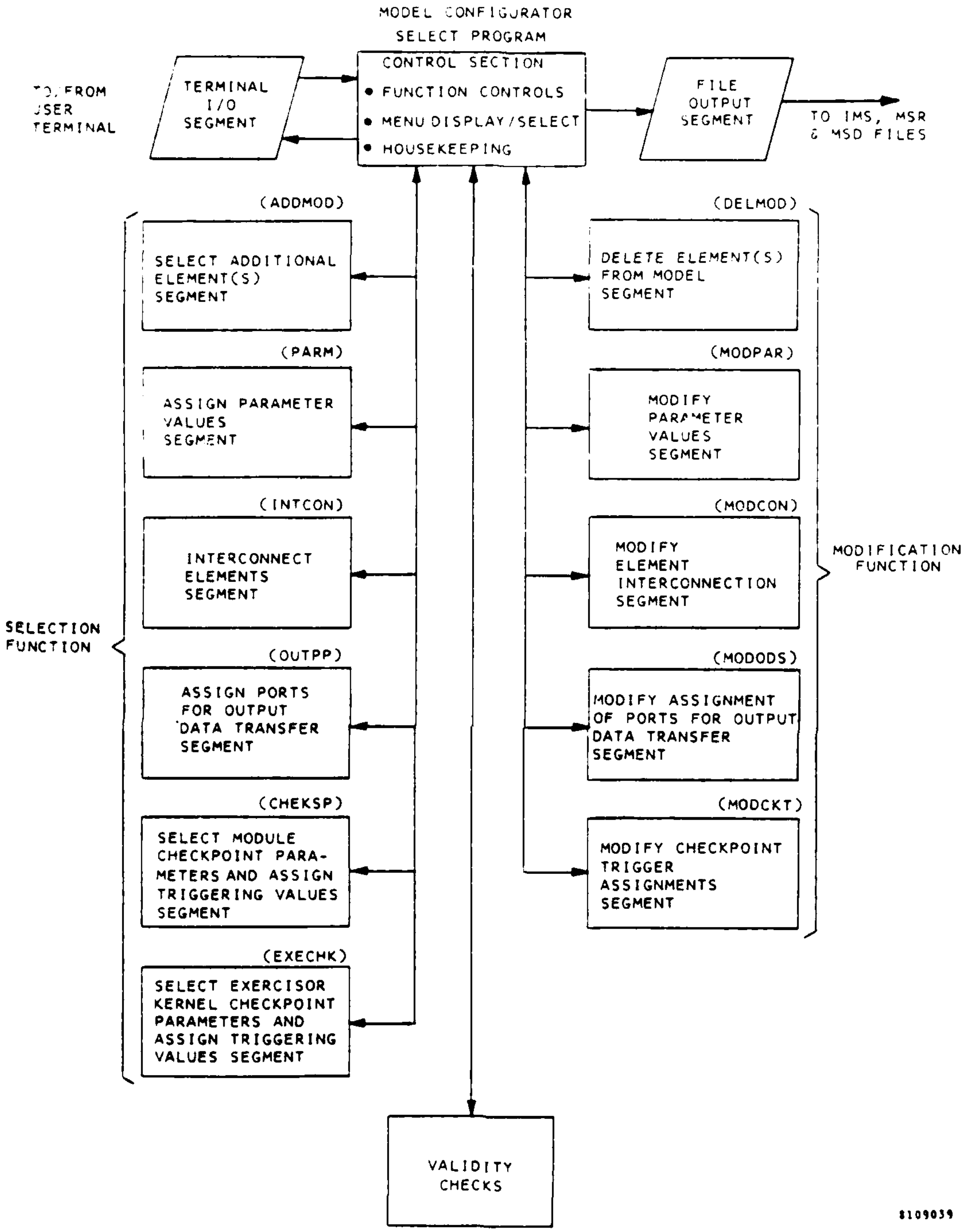


Figure 3-10. Organization of MC Select (MCS) Program

the crt terminal and facilitates a return to the previous phases of the MCS program. If a model is validated with respect to the aforementioned checks, configuration table generation and output will be performed.

The MCS program control section (root):

- o Controls switching among program phases and segments
- o Accepts crt terminal input from, and routes crt display output to, the Terminal I/O Segment
- o Provides a menu/display on the crt terminal for interaction with the user
- o Initializes program variables, performs general housekeeping functions, and provides orderly conclusion of program operation.

Upon initiation, the MCS program prompts the user to indicate if the TSM to be configured is to be executed in a batch mode or in an interactive mode of the ICSSM host computer operating system. The user will then be queried as to whether the source of the configuration input will be an existing model (specifications for which are contained in an MSR file named by the user during a previous configuration session) or whether the user will be defining a new model.

a. If the user indicates use of specifications from an MSR file, the MCS program will prompt for the name of the MSR file and then input the specifications from that file.

b. Following this preliminary interaction, the MCS program will display a menu of choices for nine different subsequent crt displays (this will be referred to as the Main Selection Menu):

(1) An interactive dialogue for guiding the user in the use of sub-model and model partitioning facilities

(2) A list of Chapters currently implemented in the LDAG from which modules may be added to the TSM configuration.

(3) A list of modules in the current TSM configuration from which modules may be deleted

(4) A list of Exercisor Kernel control segment parameters that may be selected for use as checkpoint triggers

(5) A display for interconnection specification

(6) A display for specification of output data transfer requirements specification

(7) A list of modules employed in the current model

(8) A list of modification functions to facilitiate changes in model definition

(9) A display of final validity checking messages (if any) and a query for MSR file name to be supplied by the user.

3.3.1.1.1 Sub-Models and Partitioned Models. If the first menu selection is chosen, the interactive guide for use of sub-model and model partitioning facilities is displayed. The user is instructed in procedures for configuring a sub-model or partitioned model.

3.3.1.1.2 Library Chapter Display and Module Data Input. If the second menu selection is made, Chapter browse is enabled. The program displays a numbered list of Chapters currently in the Library as well as the choice to return to the Main Selection Menu. The user is prompted to enter the number of the choice desired. If the number of the user choice corresponds to a Library Chapter, module selection from that Chapter is enabled. The program displays a menu of modules in the chosen Chapter as well as the choices to select another chapter or return to the Main Selection Menu. The user is prompted to enter the number of the desired choice. The MCS program is designed so that only one module can be chosen at a time.

a. Module Addition. Module addition is allowed when space is available in the configuration tables. Information about module functions is presented if the user selects this option. If the user indicates that a module will be used in the current configuration, parameter information for the module is retrieved and displayed, and the program prompts for the needed module parameter values. The entered values are submitted to parameter range validity checking. If any value is detected out-of-range, the MCS program issues the appropriate diagnostic to the crt and repeats the request until a valid parameter value assignment is made. If, during MCS program execution, more than 25 modules are selected, the TSM configuration tables will be filled (the

maximum number of modules in a model configuration is 25). The user is prompted to indicate whether the next action should be table consolidation, module deletion and then table consolidation, or return to the Main Selection Menu.

b. Selecting Checkpoint Parameters. Subsequent to parameter value assignments for any given module, the MCS program requests that the user select checkpoint triggers from the parameters associated with that module (providing there is space available in the checkpoint tables). If the user selects parameters to be used as checkpoint triggers, the MCS program displays pertinent information about the selected parameters and prompts the user to input triggering values for each parameter. Following checkpoint trigger selection, the program automatically enters the module input port and output port information for the selected module into the TSM "ports tables."

3.3.1.1.3 Deleting Modules. If the third menu selection is made, the module deletion function is enabled. The MCS program presents a list of modules of the current model, identified by associated module number, Library Name, and user name. The user is then prompted to indicate the total number of modules to be deleted and the associated module numbers. Upon encountering an invalid module number, the MCS program issues the appropriate diagnostic to the crt terminal. Upon completion of module deletion, the MCS program returns to the Main Selection Menu.

3.3.1.1.4 Checkpoint Control Selection. If the fourth menu selection is chosen, the Exercisor Control Segment checkpoint trigger-selection function is enabled. The MCS program displays a list of Exercisor Kernel control segment parameters that are available for use as checkpoint triggers. The user is prompted to indicate which (if any) parameters are to be used as checkpoint triggers. The user responses are checked for validity, and the MCS program issues an appropriate diagnostic to the crt terminal. The program displays pertinent information for the selected parameters and directs the user to input triggering values for each selected parameter. If at least one checkpoint trigger has been specified, the program advises the user that further selection of Exercisor Kernel checkpoint triggers must be accomplished through the Modify function, selectable via the Main Selection Menu.

3.3.1.1.5 Specification of Module Interconnection. If the fifth menu selection is chosen, the MCS program interconnection specification function is enabled. The program issues an appropriate diagnostic to the crt terminal if no input

ports are available in the present configuration and returns to the Main Selection Menu. Otherwise, the program automatically displays the current list of module input and output ports. The user is requested to input a connecting-output-port number for each input port. Connecting-output-port numbers are checked for availability and validity, and the connection data is recorded in the internal ports tables. If the user specifies an output port number that is not available, the MCS program issues an appropriate diagnostic to the crt terminal, and the user is requested to enter a valid connecting-output-port number in place of the invalid specification.

When connecting-output-ports have been assigned for all input ports in the TSM configuration, interconnection topology checks are automatically initiated.

When executing this check, the MCS program searches the ports tables for any unused ports. If any unused ports are found, an appropriate diagnostic is issued to the crt terminal. If a configuration has more output ports than input ports, the interconnection specification cannot be completed (fan-in connections are not allowed). The presence of excess output ports causes an appropriate diagnostic to be issued to the crt terminal. Following interconnection topology checks, the MCS program informs the user that any further connection data must be input using the Modify function. The program then returns to the Main Selection Menu.

3.3.1.1.6 <u>Specifying Output Data Retention</u>. If the sixth menu selection is chosen, the MCS activates a function providing specification of module output ports for data transfers. If there is at least one output port in the current TSM configuration, the user is prompted to indicate if data output from all ports is to be retained in the simulation output files. If so, the MCS program adjusts the ports table appropriately and then returns to the Main Selection Menu. If the program detects that there are <u>no</u> output module ports in the current configuration, an appropriate diagnostic is issued to the crt terminal, and the program will return to the Main Selection Menu. If the user intends to designate specific ports for output data transfers, the program displays the current output port list. At this point, the user is prompted to indicate how many ports are to be designated for data transfer and their asosciated port numbers. Port numbers input by the user are checked for availability. If the ports indicated are available, the program adjusts the ports table appropriately. If an un-

available port number is specified, an appropriate diagnostic is issued to the crt terminal. At this point, the MCS program displays the advisory message that further output data transfer specifications must be made using the Modify function. The program then returns to the Main Selection Menu.

3.3.1.1.7 Listing Modules in the Current TSM. If the seventh menu selection is chosen, the MCS program displays a listing of modules present in the current TSM configuration. If there are no modules at all in the present configuration, an appropriate diagnostic is issued to the crt terminal, and the program returns to the Main Selection Menu. If at least one module is present in the current configuration, each module is identified by its associated module number, Library name, user name, number of input ports, number of output ports, and the number of checkpoint triggers selected from the module parameters. Following the display of the module list, the MCS program returns to the Main Selection Menu.

3.3.1.1.8 Modifying an Existing TSM. If the eighth menu selection is chosen, the MCS program invokes the TSM modification function. If there were no modules in the current configuration, an appropriate diagnostic is issued to the crt terminal, and the program returns to the Main Selection Menu. If at least one module is present in the current configuration. the Model Modification Menu is displayed on the crt terminal. The model modification will provide choices for five functions:

- o Parameter value modification
- o Checkpoint trigger modification
- o Interconnection data modification
- o Output data transfer requirements modification
- o Return to the MCS program Main Selection Menu.

a. Modifying Module Parameters. If parameter value modification is selected and there are no TSM modules at all that require parameter specifications, the MCS program issues an appropriate diagnostic to the crt terminal and returns to the Model Modification Menu. Otherwise, the program displays a list of modules in the TSM configuration that requires parameters to be set. For each such module, the program displays parameter information and previously assigned values. The user is prompted to indicate those

parameters (and the numbers associated with them) that are to be modified. The user response will be checked for validity, and if an invalid selection is made, an appropriate diagnostic is issued to the crt terminal. If a valid parameter number is selected for value modification, the parameter description and range values are displayed. The user is prompted to input the modified value for the displayed parameter. If a parameter is out of range, the MCS program repeats the prompt to input a modified value within the associated valid range. Upon completion of parameter value modification, the MCS program returns to the Model Modification Menu.

b. <u>Modifying Checkpoint Triggers</u>. If checkpoint trigger modification is selected, the checkpoint trigger modification facility is enabled. A Checkpoint Modification Menu is displayed providing five choices:

- Checkpoint trigger display
- Checkpoint trigger deletion
- Checkpoint trigger value edit
- Checkpoint trigger addition
- Return to modification menu.

(1) If the first Checkpoint Modification Menu choice is made, existing checkpoint triggers for the model are displayed. If there are none, an appropriate diagnostic is displayed. The checkpoint triggers are identified by associated module number, parameter description, and triggering values.

(2) If the second Checkpoint Modification Menu choice is made, checkpoint trigger deletion is enabled. The user is prompted to indicate if the list of checkpoint triggers should be displayed. The user is then prompted to input the total number of triggers to be deleted, and their associated trigger numbers. The program adjusts the internal checkpoint trigger tables accordingly and then returns to the Checkpoint Modification Menu.

(3) If the third Checkpoint Modification Menu choice is made, the checkpoint trigger-value modification facility is enabled. If there are no checkpoint triggers specified in the TSM configuration, an appropriate diagnostic is issued to the crt terminal. If there are checkpoint triggers specified in the TSM, the user is prompted to indicate if

the list of checkpoint triggers should be displayed. The user is then prompted to indicate the total number of checkpoint triggers to be modified, and then to enter the new trigger values.

The MCS program displays the particular parameter values and the numbers associated with the checkpoint triggers. The program displays the parameter description and previous trigger value specifictions. The user is prompted to input the new triggering values for each selected checkpoint trigger.

(4) If the fourth Checkpoint Modification Menu choice is selected, checkpoint trigger addition is enabled. The user is prompted to indicate if the existing checkpoint triggers should be displayed. The MCS program displays both a list of all modules that employ settable parameters and a list of the Exercisor Kernel control segments that could be used as checkpoint triggers. The user is prompted to input the number corresponding to the module checkpoint triggers. The MCS program displays a list of parameters associated with the selected module or Exercisor Kernel segment.

The user is prompted to input the total number of parameters to be used as checkpoint triggers and the associated parameter numbers. The selected parameters are then displayed, and the user is prompted to input triggering values. After the user enters all triggering values, the MCS program returns to the Checkpoint Modification Menu.

(5) If the fifth choice from the Checkpoint Modification Menu is selected, the MCS program returns directly to the Model Modification Menu.

c. Modifying Module Interconnection Data. If the third Model Modification Menu selection is chosen, modification of TSM interconnection data is enabled. The MSC program displays the current output port list. The MCS program then displays input ports and associated connecting-output-ports. The user is prompted to input the total number of input ports that require the connecting-output/input-port numbers to be changed. If an invalid input port number is specified, an appropriate diagnostic is issued to the crt terminal. If valid input port numbers are specified, the program displays the selected input ports (identified by port numbers, associated module user name, associated module number, and input port description). The user is prompted to input connecting-output-port numbers for the selected input ports. If the responses pertain to available output port numbers, the new connection data is recorded in the

ports lists. If an unavailable output port is specified for connection to an input port, the program issues the appropriate diagnostic to the crt terminal and prompts the user to input another connecting-output-port number for the input port, until a valid number is input. After all connections data is recorded in the ports tables, the program returns to the Model Modification Menu.

d. Modifying Output Data Retention Specifications. If the fourth model modification menu choice is selected, modification of TSM simulation output data transfer requirements is enabled. If there are no output ports at all in the current TSM configuration, an appropriate diagnostic is issued to the crt terminal, and the program returns to the Model Modification Menu. If there is at least one output port in the configuration, the user is prompted to indicate if all output ports should be used for output data transfers.

3.3.1.1.9 Validity Checking. If the ninth display of the Main Selection Menu is chosen, the program enters the model completeness/consistency test and output phase. If there are no modules in the TSM configuration at this point, an appropriate diagnostic is issued to the crt terminal. After the user acknowledges the message, the MCS program returns to the Main Selection Menu. If modules are present, model testing proceeds. The TSM program searches the model configuration tables for the presence of essential modeling elements. If the model contains no self-updating module, or more than one self-updating module, or no terminating module, the MCS program issues an error message to the crt terminal. After the user acknowledges the message, the program automatically returns to the Main Selection Menu. If, however, an intermediate module is not found in the model, a warning message is issued to the crt terminal, and the user is prompted to indicate whether to return to the Main Selection Menu or to proceed with testing and output.

The MCS program tests TSM module interconnection data. The program tests internal status flags to determine if interconnection data has been specified. If no interconnections at all have been specified, the program issues an appropriate diagnostic to the crt terminal. After the user acknowledges the message, the program automatically returns to the Main Selection Menu. If connection specifications are incomplete, as indicated by a status flag, the MCS program performs connection-by-connection analysis of the connection table contents. If, following the detailed examination of the connection data, interconnection specifications prove to be incomplete, the appropriate diagnostic is issued to the

crt terminal. After the user acknowledges the message, the program automatically returns to the Main Selection Menu.

Following the interconnection topology validation, the MCS program tests output data transfer requirements. If any output ports in the current TSM configuration have been added or deleted from the model since the last assignment of output data transfer requirements, a warning is issued to the crt terminal. If the user directs the program to proceed, the MCS program searches the output port lists to find at least one port that is designated for data transfers to the output file. If no such port is found, an error message is issued to the crt terminal. The user is then prompted to select the default setting. If the user indicates that the default setting should not be used, the MCS program automatically returns to the Main Selection Menu.

Following the validation of output data transfer requirements, the MCS program prompts the user to input a file name (maximum of 10 characters) for the Model Specification Retention (MSR) file. The details of the TSM configuration are retained on the MSR for future retrieval. The program names the Model Specification Description (MSD) file by adding the prefix "doc_" to the user-specified MSR file name. The Intermediate Model Specification (IMS) file is then written, and MCS program execution terminates.

3.3.2 Description of the MC Precompiler (MCP) Program

The MCP program accepts the contents of the IMS file as input and produces seven files as outputs:

- o FORTRAN Model Description (FMD)
- o FORTRAN Common Block Alignment (FCBA) file
- o Model Table Extract (MTE)
- o TSM Checkpoint Status (TCS) file
- o FORTRAN Checkpoint Trigger Controller (CTC)
- o FORTRAN Output Data Transfer Controller (OTC)
- o Module and Output Port (MOP) file

The IMS file is read sequentially, and information destined for the output files is extracted from the records thus read, through the internal processing of the MCP program.

The FMD file contains a model-dependent version of the Exercisor Simulator Process (ESP) Module (subroutine PROCES). The ESP is the application-specific modeling segment that, in conjunction with the ESK, makes up an individual TSM.

Each communication system model submitted to ICSSM for simulation gives rise to a specific version of the ESP/EK combination that constitutes the TSM.

The FMD file contains an automatically generated FORTRAN version of the ESP that, after submission to the FORTRAN Compiler, is included in the TSM.

The MTE file contains tables that record module interconnection data and control data used to regulate and direct the TSM during the course of execution. The tables consist of:

- o Master Module List - a list of modules selected for the TSM along with their pertinent parameter and control data
- o Node List - a list of connection nodes that exist in the TSM
- o Parameter List - a list of parameter values associated with each of the modules comprising the TSM
- o To-List - a list of module interconnection data that specifies the TSM topology
- o Checkpoint Trigger Value List - a list of values upon which associated checkpoint triggers will operate.

When employed with the MULTICS system, the MCP program also generates the FCBA file, containing automatically composed MULTICS SFC commands. The FCBA file is generated in compliance with the requirement (when executing FORTRAN programs under the MULTICS operation system) to perform in-main-storage alignment of FORTRAN Common blocks among a FORTRAN mainline and its CALLed subroutines. The SFC commands required differ from TSM to TSM, as different application modules are expected to be selected for use in different TSMs.

The MCP program outputs an initial value into the TCS file. The Exercisor Kernal references the TCS file. The initialized value indicates to the Exercisor Kernal that no checkpoint event has occurred and therefore the TSM execution should be initiated.

The CTC and the OTC files contain FORTRAN program segments that are joined to the TSM to provide the user-specified checkpoint and data output functions during simulation execution.

The FMD and MTE files provide data-input to the Exercisor/ Simulator subsystem and are identified to the ICSSM host computer operating system as the PROCES.FORTRAN file and the ETBL_DAT file, respectively.

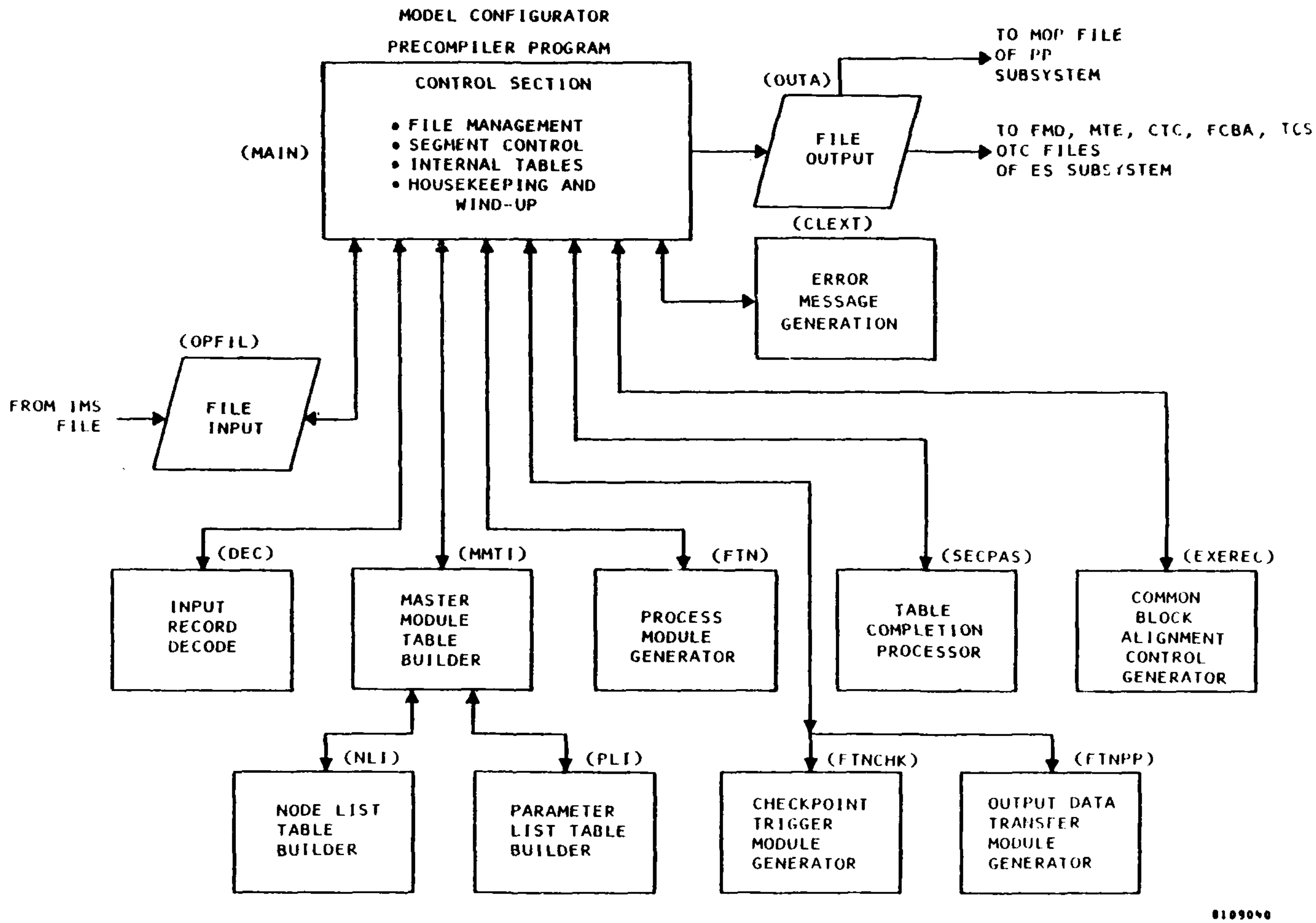


Figure 3-11. Organization of MC Precompiler (MCP) Program

The MOP file contains a table of modules to be used in the TSM along with associated topological parameters of the TSM. The MOP is used as an input to the Post-Processor subsystem.

The MOP file contains data for input to the Post-Processor subsystem and is identified to the ICSSM host computer operating system as the PTBL_DAT file.

3.3.2.1 Major Operations Performed in the MCP Program

A macro-level flowchart depicting the functional operation for the MCP program is provided in figure 3-11.

The MCP program operates with a two-pass structure: Pass A reads all records offered as input on the IMS file, checks the record syntax and topological consistency of module connection information found there, and builds internal tables requires in the formation of the MTE and MOP files.

Pass B completes the table-building process and constructs the FMD, OTC, FCBA, TCS, and CTC files

The MCP employs program segments as shown in figure 3-11. The various segments indicated employ a FORTRAN subroutine structure anduse suggestive subroutine names as indicated. The charts of figures 3-12 and 3-13 indicate the functional flow among the various program segments.

Error checking in the MCP program is performed within program segments logically associated with anticipated error conditions. The IMS file is produced by the MCS program, which has its own error checking/validation facilities. Any error that is checked in the MCS program is not re-checked by the MCP program. The Precompiler issues error messages to the crt terminal. The error messages, organized with respect to the Precompiler program segments in which they are expected to occur, are defined in table 3-2.

The organizational hierarchy of the MCP program is depicted in figure 3-14.

Table 3-2. Precompiler Program Error Checks

In Constructing MTE File Tables

Error #1 - "Size of the Node List has exceeded size of working array."

Error #2 - "Size of Parameter List has exceeded program limits."

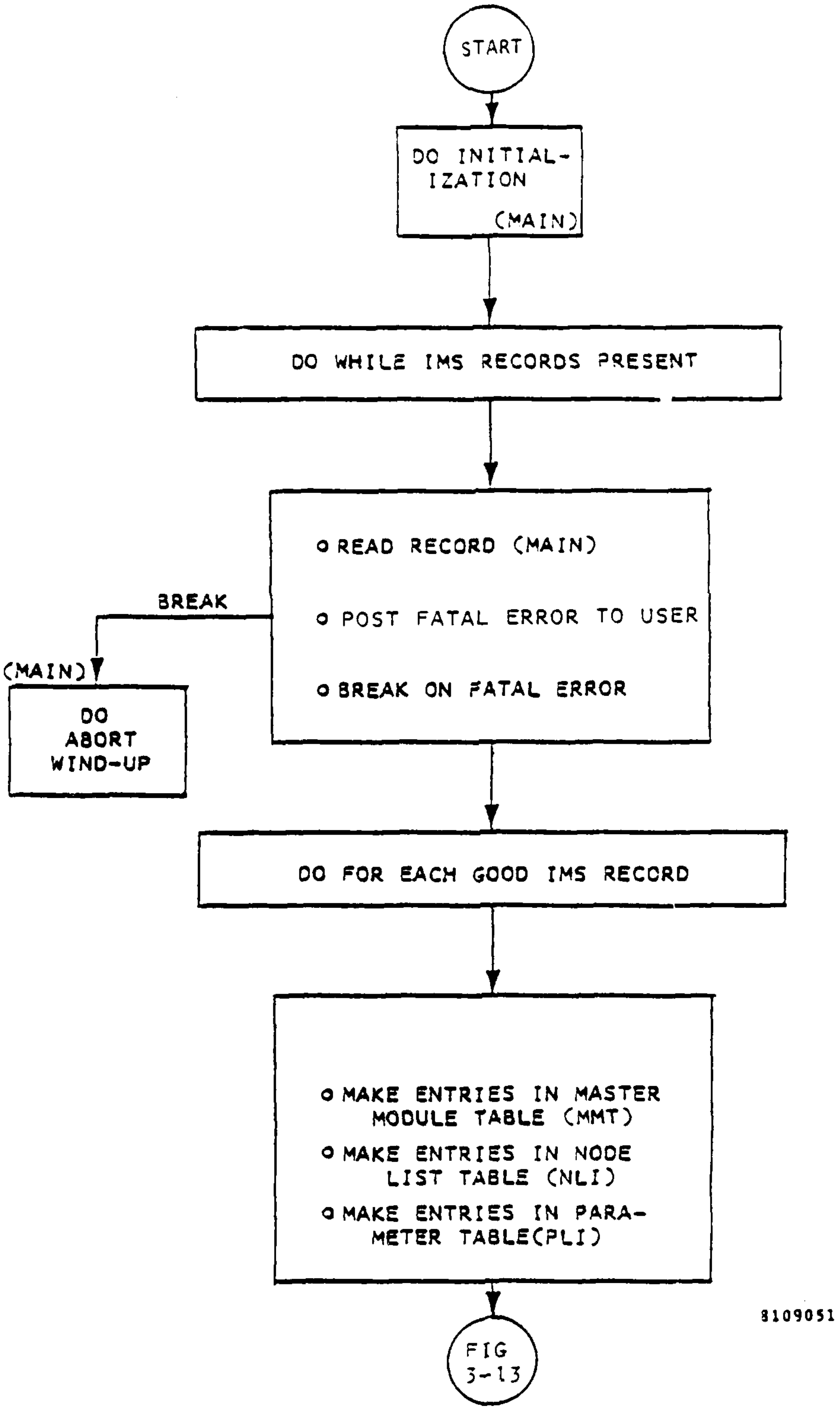


Figure 3-12. MC Precompiler (MCP) Program - Functional Flow (Pass A)

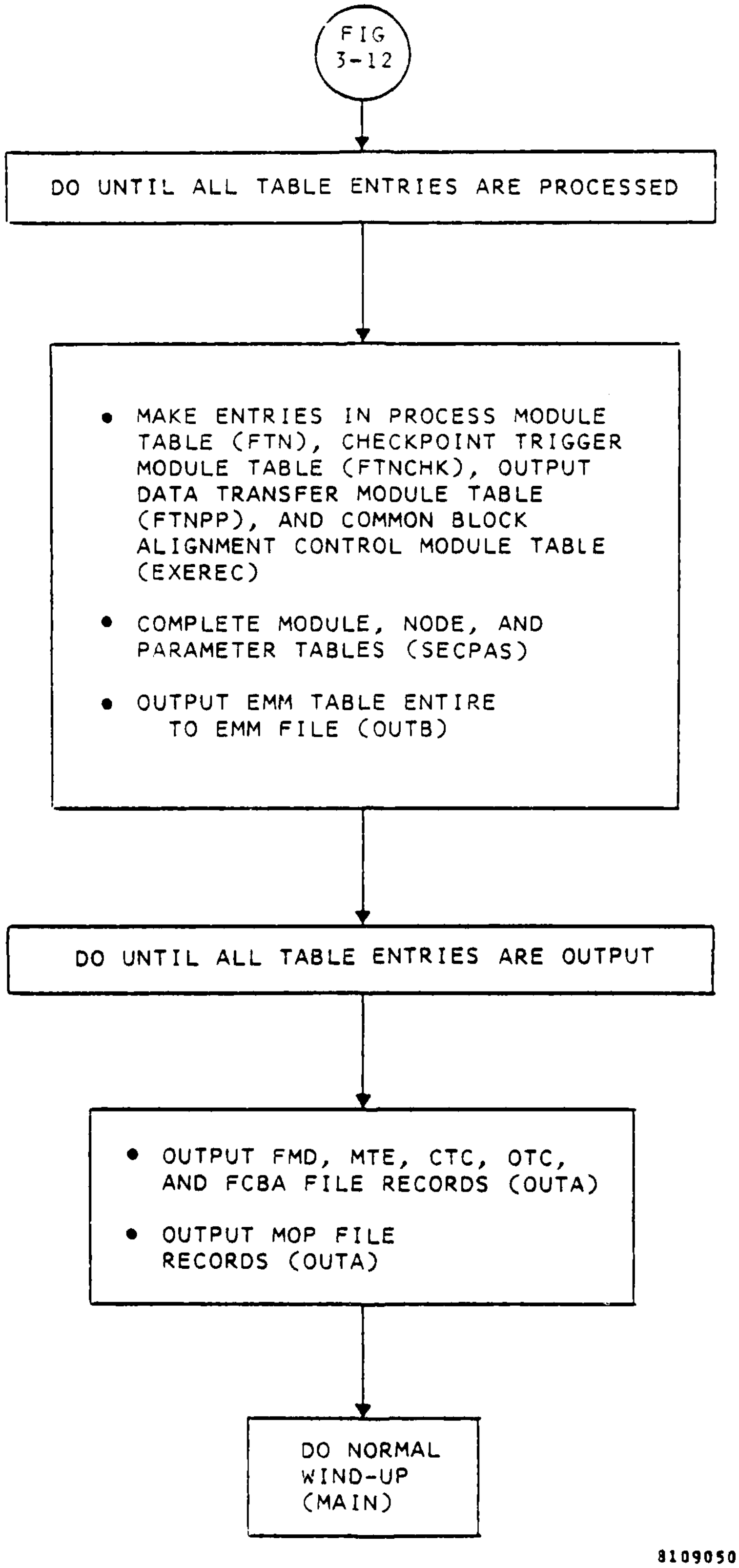


Figure 3-13. MC Precompiler (MCP) Program - Functional Flow (Pass B)

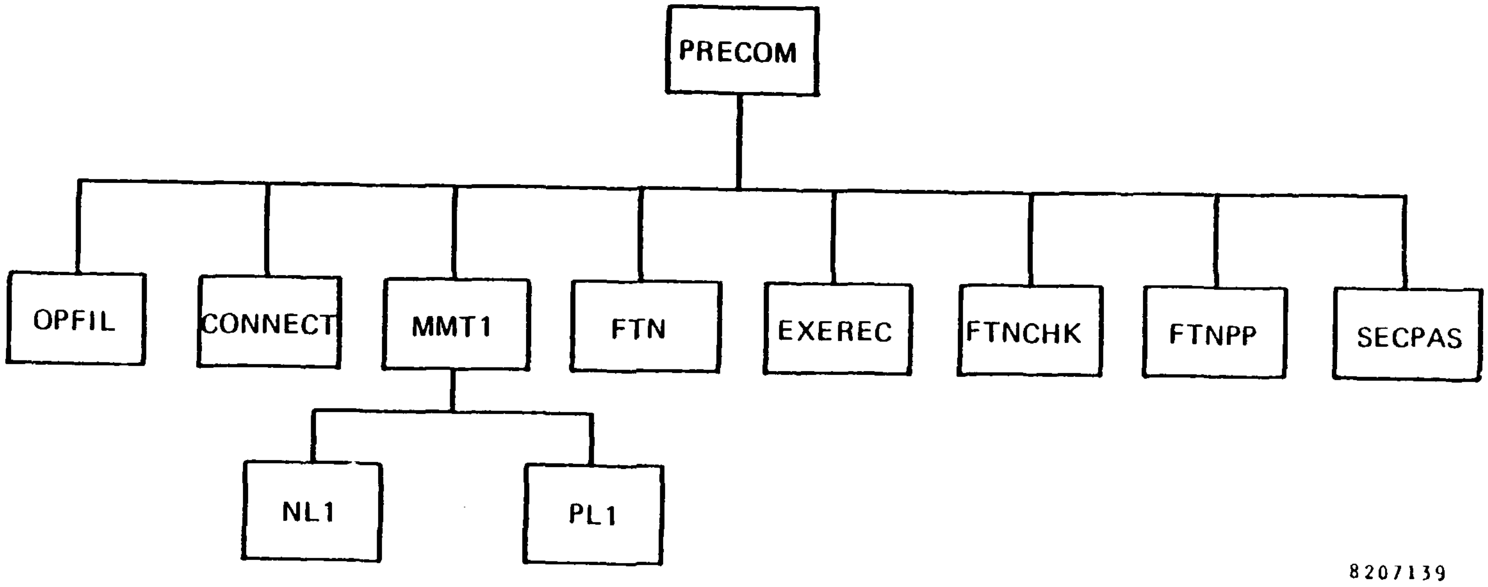


Figure 3-14. MCP Program Hierarchy

Table 3-2. Precompiler Program Error Checks (Continued)

In Completing List Connections and To-List

Error #1 - "Connecting module and port don't exist." (This error implies that table of modules and/or associated ports from IMS input file contains inconsistent data.)

Error #2 - "Internal pointer value exceeds To-List array size." (This error implies that too many connections have been defined, and list will exceed assigned array size.)

3.3.3 Description of the General Target Simulation Model (TSM) Program

The ES subsystem provides the facilities to compose a TSM and to control TSM execution. A typical TSM consists of a fixed (ie, programmatically constant) mainline (viz, the EK) and an automatically written, customized simulator module (viz, the PM) linked to it to form the complete customized TSM.

The elements that comprise the ES subsystem are shown in figure 3-5. The FMD produced by the MC subsystem is submitted to the ICSSM host computer FORTRAN Compiler. By this mechanism, the model-specific PROCESS module is created in linkable, executable form, for incorporation into the TSM. In addition, the EK (FORTRAN-coded but submitted previously to the FORTRAN Compiler and available in a linkable, executable form) is invariant in every TSM.

The CTC and OTC files produced by the MC subsystem are submitted to the ICSSM host computer FORTRAN Compiler. The TSM uses the CTC file to control the scheduling of checkpoint events. The OTC file is used to control data transfers from the ICSSM system to the output files.

The TSM uses the MTE file as input and produces three **(two optional)** files as output: (For more details see DATA REQUIREMENTS Document, report 6451)

- Execution Journal (EXJ) **(Optional)**
- Signal Output (SIG)
- Event Journal (EVJ) **(Optional)**

The TSM also uses the temporary I/O files: Signal Work (SW), Coefficient Work (CW), Signal List Checkpoint Status (SCS) and Coefficient Checkpoint Status (CCS).

Program termination in the TSM program is automatic. When TSM program termination occurs, the EVJ file contents may be examined to determine if errors were detected during TSM program execution.

3.3.3.1 Major Operations Performed in the TSM Program

Organization and major functions in the TSM Program are depicted in figures 3-15 and 3-16. The individual macro-functions depicted are delineated in separate macro-level charts as:

- o Control Table Management Portion - figure 3-17a.
- o Event Queue Table Processor Portion - figure 3-17b.
- o Process Module Portion - figure 3-18.

The names in parentheses in these figures are FORTRAN subroutine names used within the TSM.

3.3.3.1.1 Operations Performed in the TSM, Control/Executive (CE) Portion. The TSM program, CE portion:

a. Reads the contents of the MTE file and transfers the values provided by its data to in-main-storage tables corresponding to those of figures 3-19 through 3-25.

b. Initializes data for the Event Queue table and the other in-main-storage tables.

c. Processes Event Queue entries according to given rules and maintains the Event Queue in proper order as relevant occurrences within the simulation are encountered.

d. Controls the transfer of Signal and Coefficient data blocks to and from SW and CW files and to the SIG file.

e. Oversees the transfer of data and control elements to and from the Process Module Portion.

3.3.3.1.2 Operations Performed in the TSM, Control Table Management (CTM) Processor Portion. The TSM program, CTM portion operates as indicated in figure 3-16. The CTM portion:

a. Provides a means of controlling the interface between the CE portion and TSM control tables.

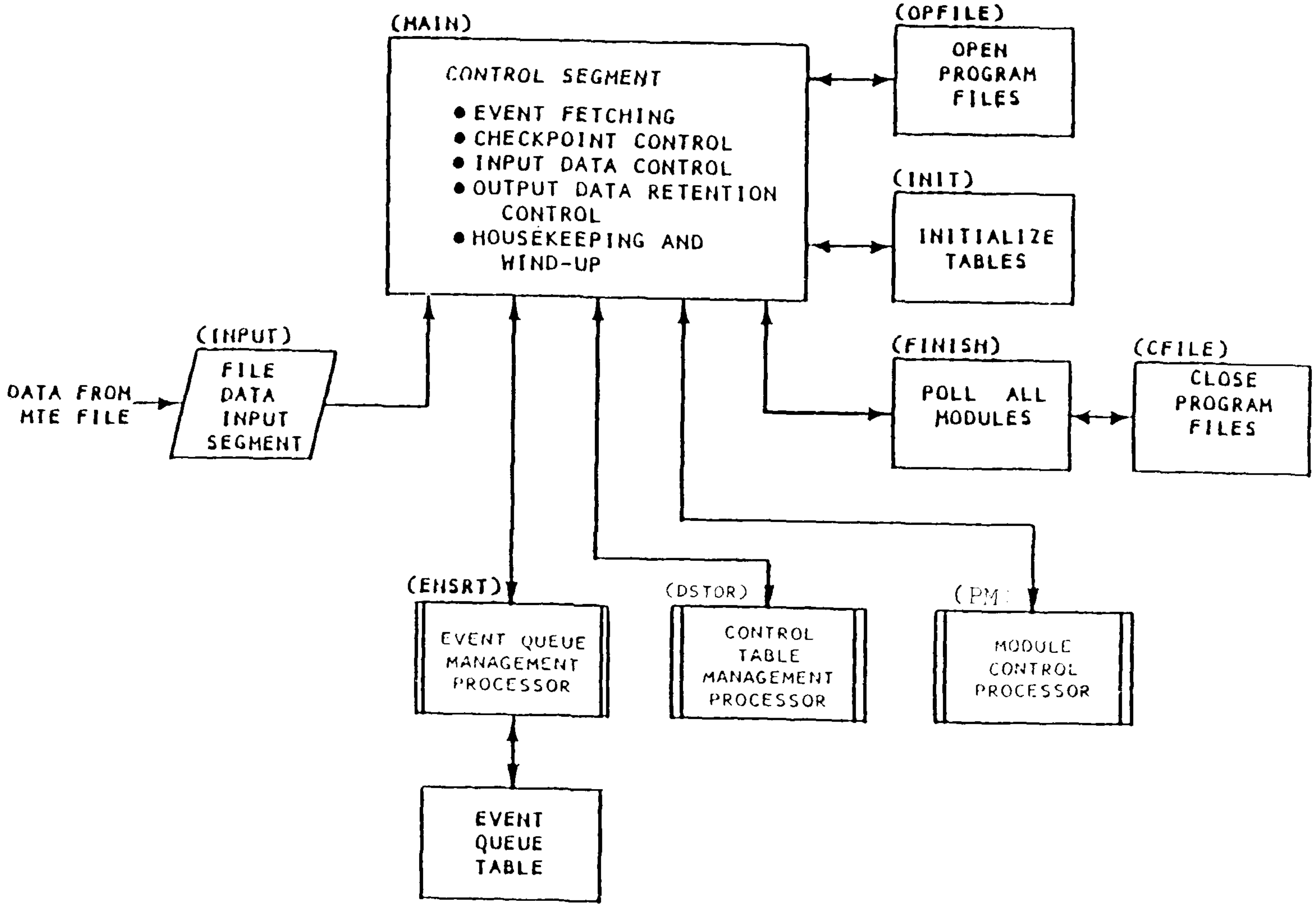


Figure 3-15. Organization of TSM Program, Control/Executive (CE) Portion

AD-A135 704 INTERACTIVE COMMUNICATION SYSTEMS SIMULATION MODEL (ICSSM) EXTENSION(U) HAZELTINE CORP GREENLAWN NY I GERRY ET AL. JUL 83 6479 RADC-TR-83-165 2/2

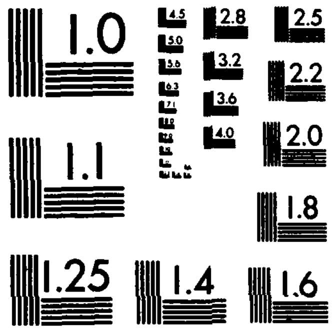
UNCLASSIFIED F30602-81-C-0001 F/G 17/2 NL

END

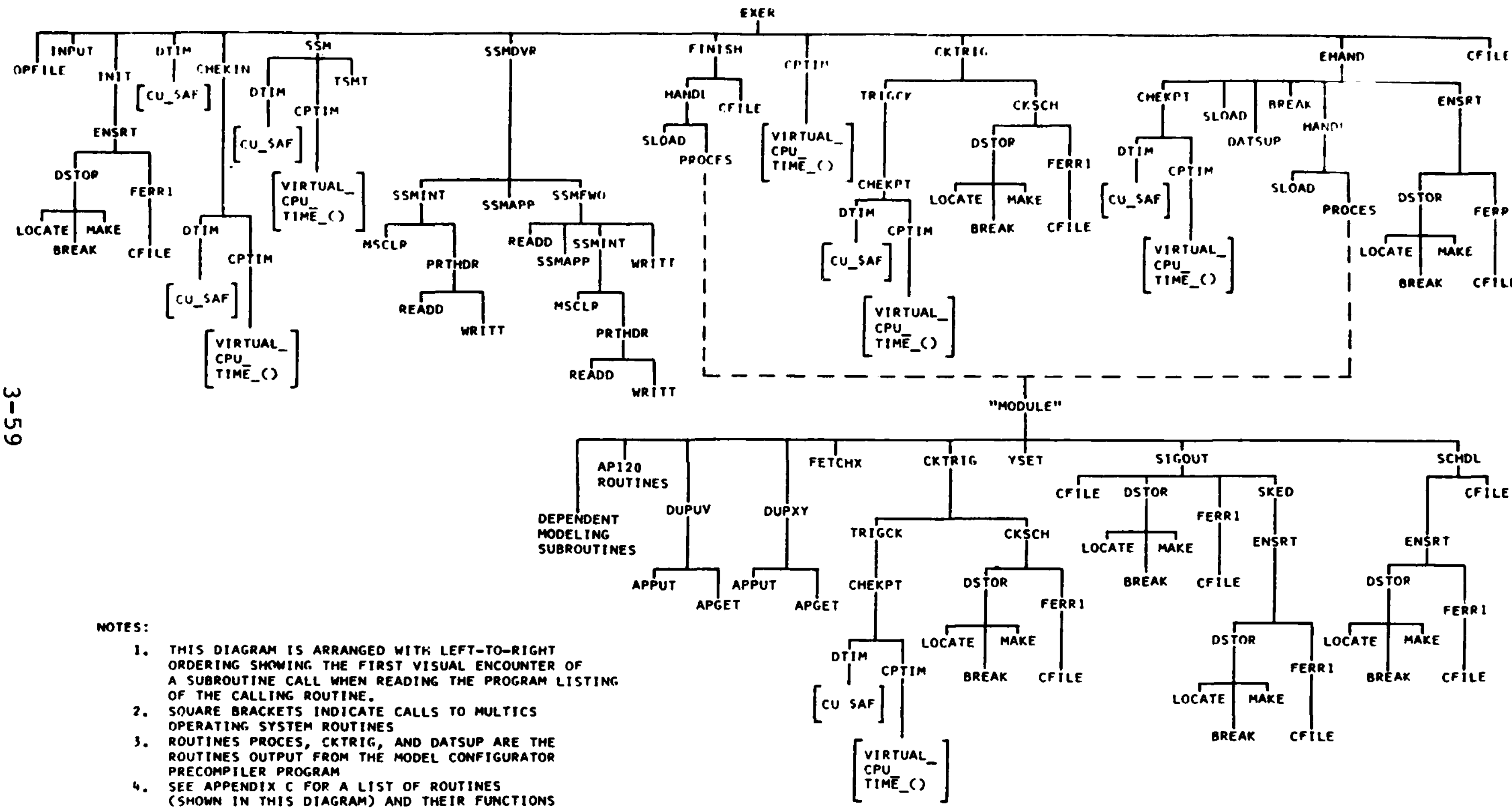
FILMED

1-84

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



NOTES:

1. THIS DIAGRAM IS ARRANGED WITH LEFT-TO-RIGHT ORDERING SHOWING THE FIRST VISUAL ENCOUNTER OF A SUBROUTINE CALL WHEN READING THE PROGRAM LISTING OF THE CALLING ROUTINE.
2. SQUARE BRACKETS INDICATE CALLS TO MULTICS OPERATING SYSTEM ROUTINES
3. ROUTINES PROCES, CKTRIG, AND DATSUP ARE THE ROUTINES OUTPUT FROM THE MODEL CONFIGURATOR PRECOMPILER PROGRAM
4. SEE APPENDIX C FOR A LIST OF ROUTINES (SHOWN IN THIS DIAGRAM) AND THEIR FUNCTIONS

Figure 3-16. TSM Subroutine Calling Hierarchy

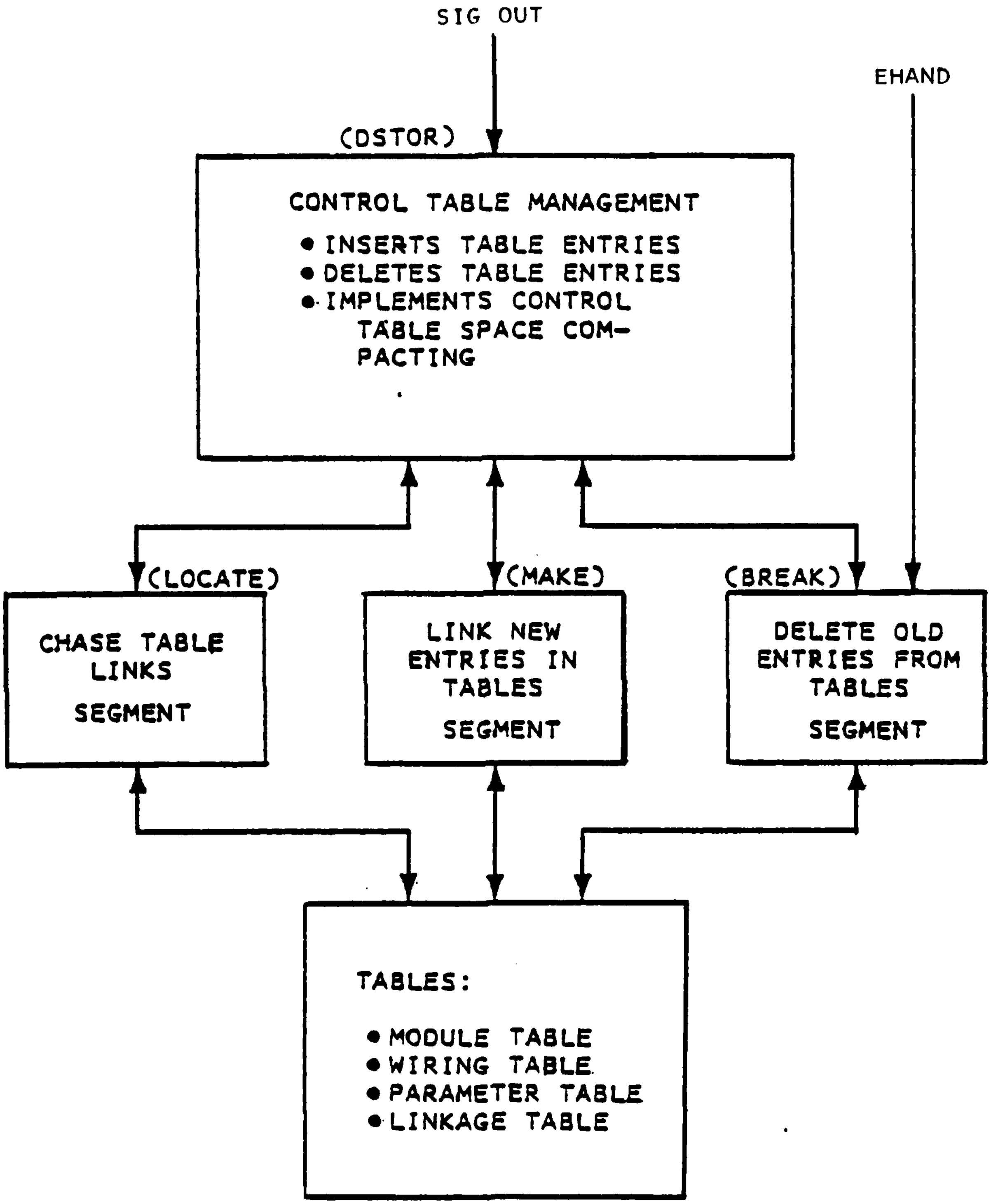


Figure 3-17a. Organization of the Control Table Management Processor

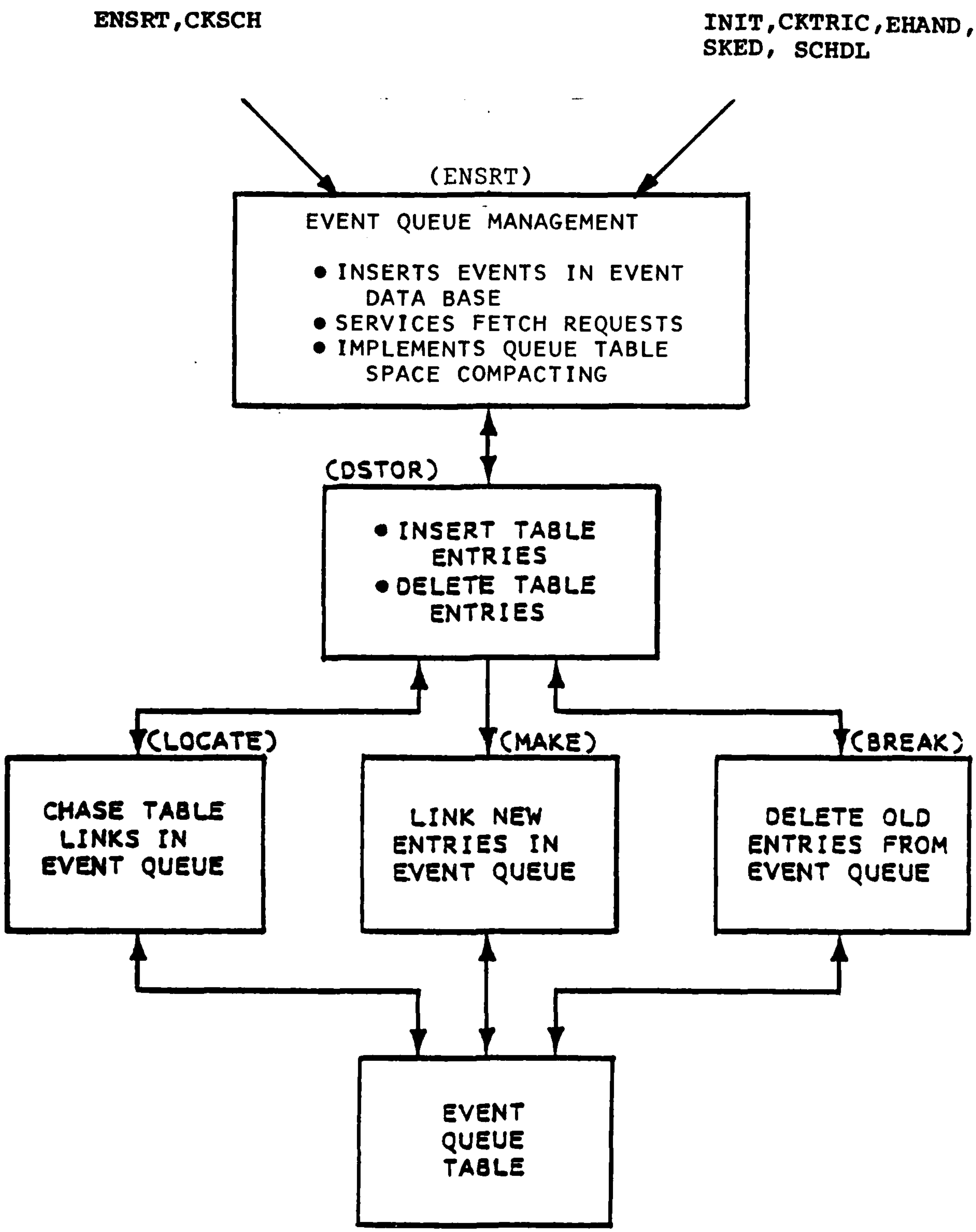


Figure 3.17b. Organization of Event Queue Management Processor

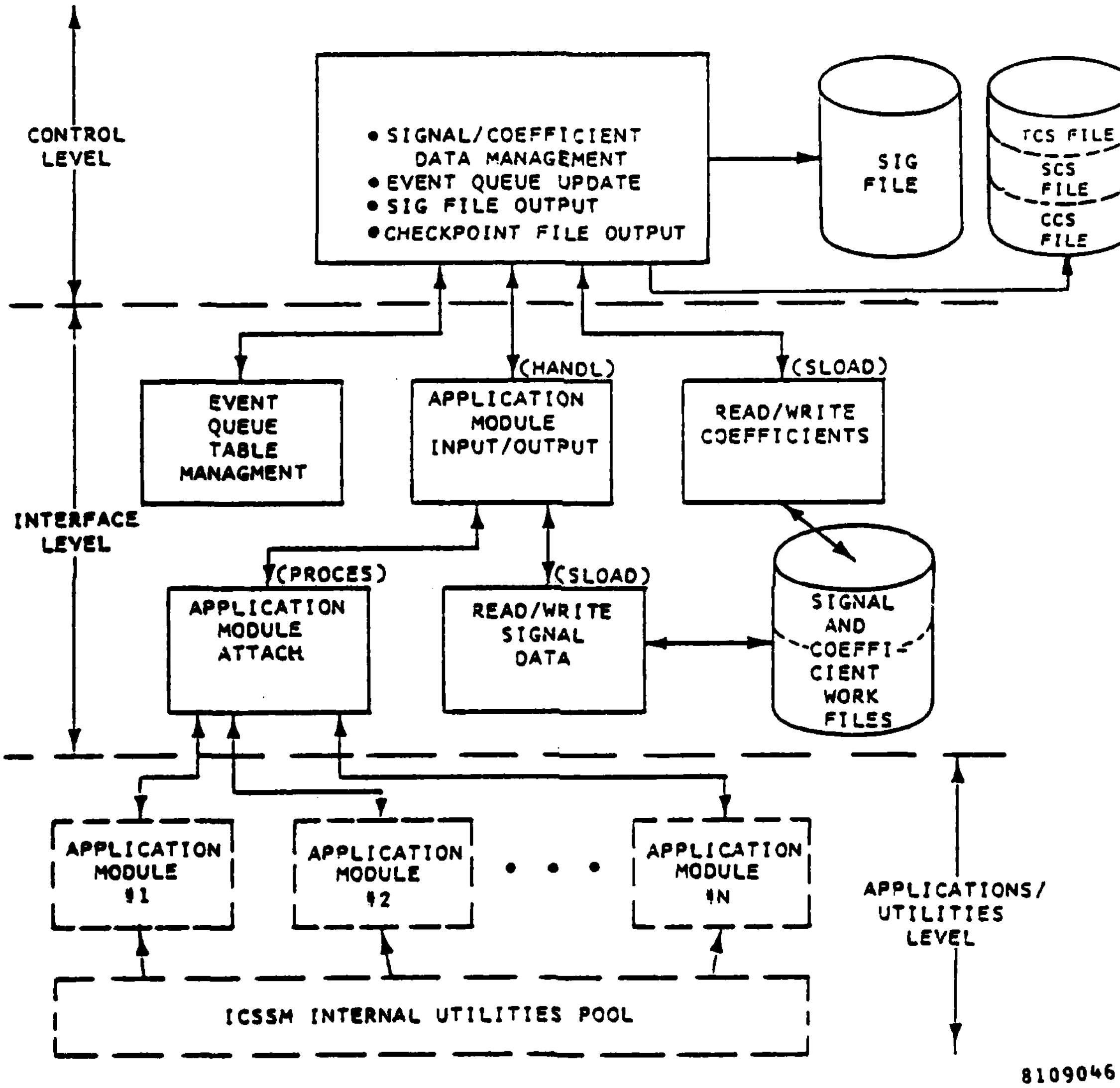


Figure 3-18. Organization of TSM Program, Process Module (PM) Portion

ADDRESS (IMPLICIT)	BACKWARD THREAD POINTER	FORWARD THREAD POINTER	QUEUE ORDERING (TIME/EVENT/NODE)	MEMBERSHIP LINKING POINTER
	QA	QB	QC	QD
1	0	2	T_1	4
2	1	3	T_2	9
3	2	101	T_3	50
4	0	5	E_{11}	7
5	4	6	E_{12}	51
6	5	102	E_{13}	52
7	0	8	N_{111}	CRP_1
8	7	103	N_{112}	CRP_2
9	0	10	E_{21}	11
10	9	103	E_{22}	53
11	0	104	N_{211}	CRP_{99}
12				

Figure 3-19. Organization of the Event Queue Table (Common: ELTST; Variables: LANKA RLSTA, IAUAX).

A	B	C	D	E	F	G	H	J	K	L
1	A121	SRG	1	2	0	0	1	2	1	4
2	A017	ENT	3	3	3	1	4	2	5	3
3	A999	DEC	6	7	6	2	8	1	8	3
4	A181	PHL	13	2					11	4
5										
6										

MODULE IDENTITY (A–C) · MODULE PARAMETERS (REAL) (D–E) · MODULE INPUTS (F–G) · MODULE OUTPUTS (H–J) · MODULE PARAMETERS (INTEGER) (K–L)

MODULE TABLE

A. MODULE NUMBER (IMPLIED BY POSITION IN TABLE)

B. LIBRARY-ASSIGNED MODULE NAME

C. USER-ASSIGNED MODULE NAME

D. POINTER TO HEAD OF PARAMETER SET IN PARAMETER TABLE FOR GIVEN MODULE (REAL)

E. NUMBER OF MEMBERS OF PARAMETER SET FOR GIVEN MODULE (REAL)

F. POINTER TO HEAD OF NODE-CODE SET(INPUT NODES) IN NODE TABLE FOR GIVEN MODULE

G. NUMBER OF MEMBERS IN NODE-CODE SET(INPUT NODES) FOR GIVEN MODULE

H. POINTER TO HEAD OF NODE-CODE SET (OUTPUT NODES) IN NODE TABLE FOR GIVEN MODULE

J. NUMBER OF MEMBERS IN NODE-CODE SET (OUTPUT NODES) FOR GIVEN MODULE

K. POINTER TO HEAD OF PARAMETER SET IN PARAMETER TABLE FOR GIVEN MODULE (INTEGER)

L. NUMBER OF MEMBERS OF PARAMETER SET FOR GIVEN MODULE (INTEGER)

8109049

Figure 3-20. Organization of the Module Table (Common: MODLST; Variable: MMT).

M	N	O	P	Q	R
1	1	1	1	1	2
2	1	2	1	2	4
3	2	1	1	0	1
4	2	2	2	3	9
5	2	3	1	5	5
6	3	1	-1	0	2
7	3	2	-1	0	5
8	3	3	1	7	12
9	4	1	-1	0	8

NODE TABLE

M. NODE CODE (IMPLIED BY POSITION IN TABLE)

N. MODULE NUMBER

O. PORT NUMBER

P. NUMBER OF CONNECTIONS DEFINED FOR MODULE PORT (NEGATIVE VALUE SIGNIFIES PORT IS AN INPUT PORT)

Q. POINTER TO BEGINNING OF MODULE-PORT-CONNECTED-TO THREAD IN TO-LIST TABLE

R. MODULE PORT ASSOCIATION POINTER [IF MODULE PORT IS AN OUTPUT PORT, POINTS TO BEGINNING OF MODULE PORT SIGNAL-LIST-LOCATIONS THREAD IN SIGNAL/COEFFICIENT LOCATOR TABLE. IF MODULE PORT IS AN INPUT PORT, CONTAINS NODE CODE OF PORT FROM WHICH THE PORT CONNECTION AROSE]

8109048

Figure 3-21. Organization of the Node Table (Common: MODLST; Variable: NL).

T	S
1	3
2	6
3	17
4	20
5	7
6	10
7	9
8	
9	
10	

S. CONTAINS NODE CODE WHICH MAY BE USED AS POINTER TO FIND POSITION IN NODE TABLE AT WHICH CONNECTED-TO-MODULE/ PORT INFORMATION IS LOCATED

T. ADDRESS (IMPLIED) OF MODULE-PORT-CONNECTED-TO NODE-CODE. THIS ADDRESS IS USED IN COLUMN P OF NODE TABLE

NOTE: THE VALUE CONTAINED IN COLUMN N OF NODE TABLE, IF GREATER THAN 1, DEFINES THE NUMBER OF SEQUENTIAL TO-LIST TABLE ENTRIES WHICH ARE NEEDED TO DESCRIBE FAN-OUT CONNECTIONS SO SIGNIFIED.

8109045

Figure 3-22. Organization of the To-List Table (Common: MODLST; Variable: ITO).

U	V
1	P_{11}
2	P_{12}
3	P_{21}
4	P_{22}
5	P_{23}
6	P_{31}
7	P_{32}
8	P_{33}
9	P_{34}
10	P_{35}
11	P_{36}
12	P_{37}
13	P_{41}
14	P_{42}

W	X
1	I_{11}
2	I_{12}
3	I_{13}
4	I_{14}
5	I_{21}
6	I_{22}
7	I_{23}
8	I_{31}
9	I_{32}
10	I_{33}
111	I_{41}
12	I_{42}
13	I_{43}
14	I_{44}

U. ADDRESS (IMPLIED) OF PARAMETER VALUE ASSOCIATED WITH MODULE IN MODULE TABLE. THIS ADDRESS IS USED IN COLUMN D OF MODULE TABLE. (REAL PARAMETERS)

V. A PARAMETER VALUE (REAL)

W. ADDRESS (IMPLIED) OF PARAMETER VALUE ASSOCIATED WITH MODULE IN MODULE TABLE. THIS ADDRESS IS USED IN COLUMN K OF MODULE TABLE. (INTEGER PARAMETERS)

X. A PARAMETER VALUE (INTEGER)

8109044

Figure 3-23. Organization of the Parameter Table (Common: MODLST; Variables: PL, IPL)

Y	Z	α	β	γ
1	9	13	T_{32}	12
2	0	3	T_{11}	1
3	2	7	T_{12}	4
4	0	6	T_{21}	2
5	0	8	T_{21}	5
6	4	10	T_{42}	3
7	3	0	T_{13}	6
8	5	0	T_{22}	8
9	0	1	T_{31}	10
10	6	0	T_{51}	7

SIGNAL LOCATOR

Φ	θ	δ
1	0	4
2	0	3
3	2	7
4	1	6
5	0	8
6	4	0
7	3	0
8	5	0
9	14	15
10	0	12

COEFFICIENT LOCATOR

Y. ADDRESS (IMPLIED) OF HEAD OF SIGNAL-RECORD THREAD. ALSO IS RECORD NUMBER FOR SIGNAL WORK FILE. USED AS POINTER IN COLUMN Q OF NODE TABLE.

Z. POINTER TO PREVIOUS BEAD ON SIGNAL-RECORD THREAD

α. POINTER TO NEXT BEAD ON SIGNAL-RECORD THREAD

β. SIMTIME VALUE ASSOCIATED WITH SIGNAL-RECORD AT TABLE ADDRESS. COLUMN Y VALUES ARE IN MONOTONIC NON-DECREASING ORDER FOR EACH THREAD

γ. ADDRESS OF COEFFICIENT LOCATOR TABLE ENTRY α FOR COEFFICIENT RECORD ASSOCIATED WITH SIGNAL-RECORD. ALSO IS RECORD NUMBER FOR COEFFICIENT WORK FILE

Φ. SIMILAR TO Y, BUT FOR COEFFICIENT RECORDS.

θ, δ. SIMILAR TO Y, Z, BUT FOR CO-EFFICIENT-RECORD THREAD

8109053

Figure 3-24. Organization of Signal/Coefficient Location Table (Common: SLIST; Variables: LNKB, RLSTB, NLSTB and Common: CLIST; Variable: LNKC).

CHECKPOINT TRIGGER NUMBER (IMPLIED)	LOWER TRIGGERING VALUES
1	LT_1
2	LT_2
3	LT_3
4	LT_4
5	LT_5
6	LT_6
7	LT_7

CHECKPOINT TRIGGER LOWER BOUNDARY TABLE

CHECKPOINT TRIGGER NUMBER (IMPLIED)	UPPER TRIGGERING VALUES
1	UT_1
2	UT_2
3	UT_3
4	UT_4
5	UT_5
6	UT_6
7	UT_7

CHECKPOINT TRIGGER UPPER BOUNDARY TABLE

CHECKPOINT TRIGGER NUMBER (IMPLIED)	TRIGGER INCREMENTAL VALUES
1	TI_1
2	TI_2
3	TI_3
4	TI_4
5	TI_5
6	TI_6
7	TI_7

CHECKPOINT TRIGGER INCREMENTAL VALUE TABLE

8110232

Figure 3-25. Organization of the Checkpoint Table (Common: CHEKTR; [illegible]OTRG, RUPTRG, TRGINC).

b. Provides a means to locate and retrieve records in SW and CW files that are pertinent to the Event Code type and associated Node code for the "energized" applications module invoked by the Event Queue/Event Queue Processor action.

c. Inserts new entries into the relevant control tables based upon internal action of the "energized" applications module.

d. Deletes entries from the relevant control tables based upon internal action of the "energized" applications module.

3.3.3.1.3 Operations Performed in the TSM, Event Queue Table Processor (EQP) Portion. The TSM program, EQP portion:

a. Inserts and deletes events into and from the Event Queue table.

b. Maintains a minimum inventory of empty or unused Event Queue table locations.

c. Interfaces to the TSM-CW event-fetch segment and services the fetch request.

3.3.3.1.4 Operation Performed in the TSM, Process Module (PM) Portion. The TSM program, PM portion operates on three levels:

- Control
- Interface
- Applications/Utilities.

The Control Level provides:

a. Interface to the TSM, CE portion.

b. Output of signal and coefficient records to the SIG file from the CW and SW files.

c. Interface to the EQP portion whenever an "energized" applications module issues an "update" request.

d. Communication with the Interface-level segments of the PM.

The Interface Level provides:

a. Control of Signal List and Signal/Coefficient data transfer between applications/utilities-level segments and TSM-CW COMMON areas.

b. Transfer of state-parameter value and model-specified parameter values to and from the "energized" applications module via the applications/utility-level segments.

The Applications/Utilities Level provides:

a. Subroutine link and data/parameter processing, which is manifested in the applications library modules "attached" via the customized PROCESS subroutine.

b. Utility subroutine support from subroutines used in common among applications module algorithms to effect commonly required control and data manipulations.

The relationships that prevail for Control-level/Interface-level coupling are depicted in figure 3-26.

The functional relations that prevail for Interface-level/Applications-level coupling are depicted in figure 3-27.

The TSM program internal utility functions (available from the ICSSM LDUG) are depicted in figure 3-28.

3.3.3.2 Internal Tables Used in the TSM Program

TSM uses eight internal tables:

- Event Queue
- Module
- Node
- To-List
- Real Parameters
- Integer Parameters
- Signal/Coefficient Location
- Checkpoint

The Event Queue table is processed by the Event Queue Processor (ENSAT). The remaining tables are processed by the Control Table Management Processor (DSTOR).

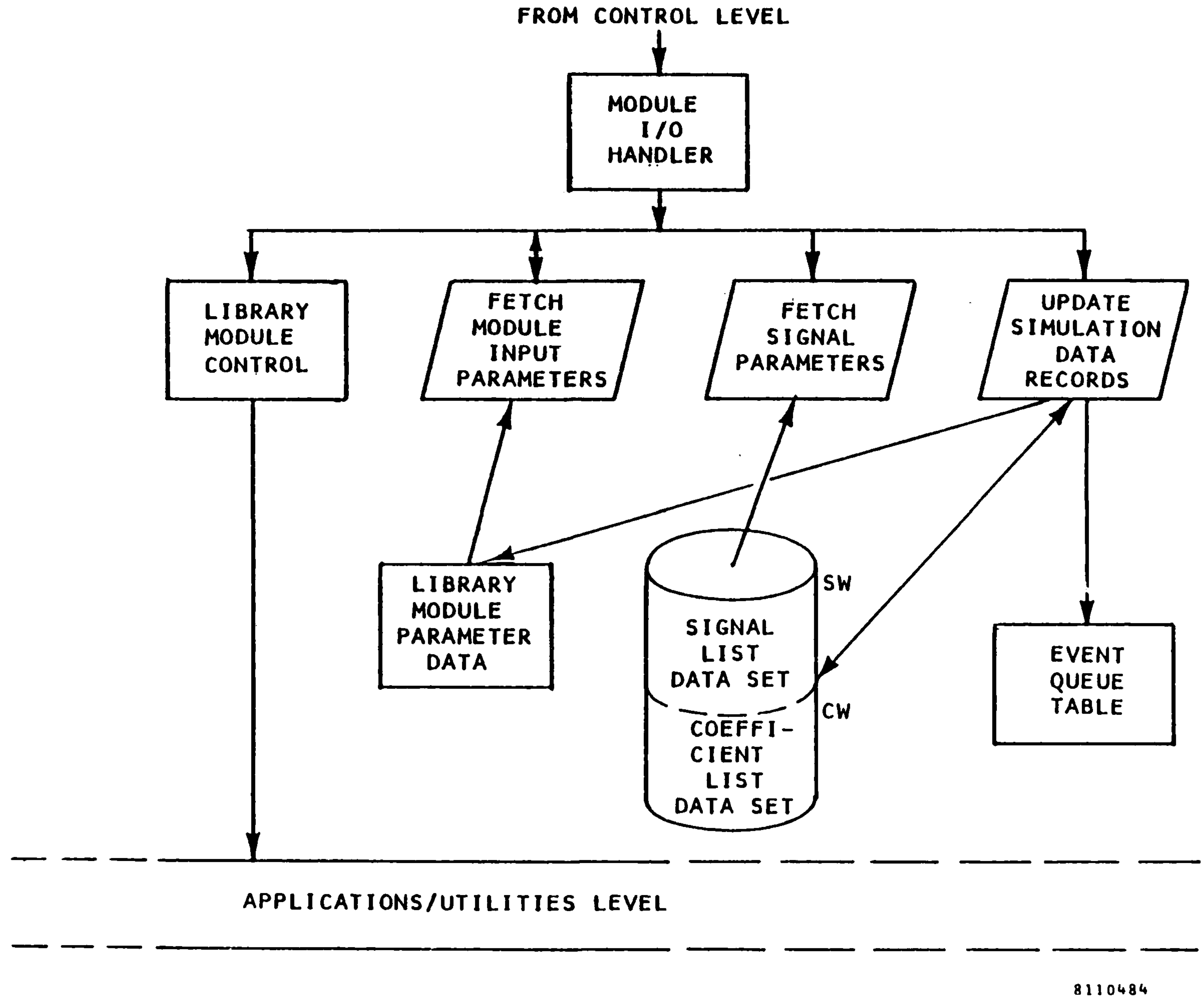


Figure 3-26. Control-Level/Interface-Level Processing, in Process Module

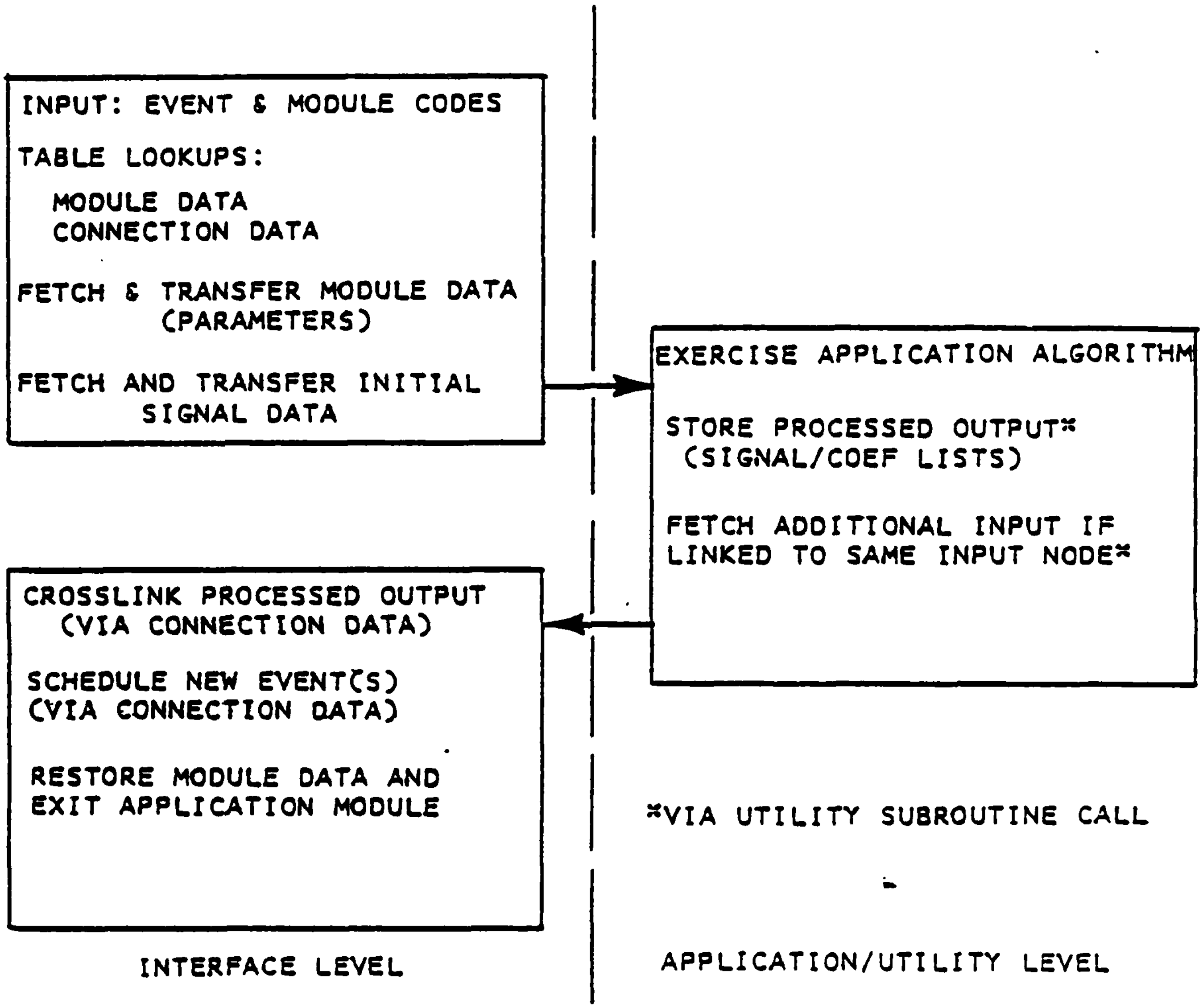


Figure 3-27. Interface-Level/Applications-Level Coupling, in Process Module

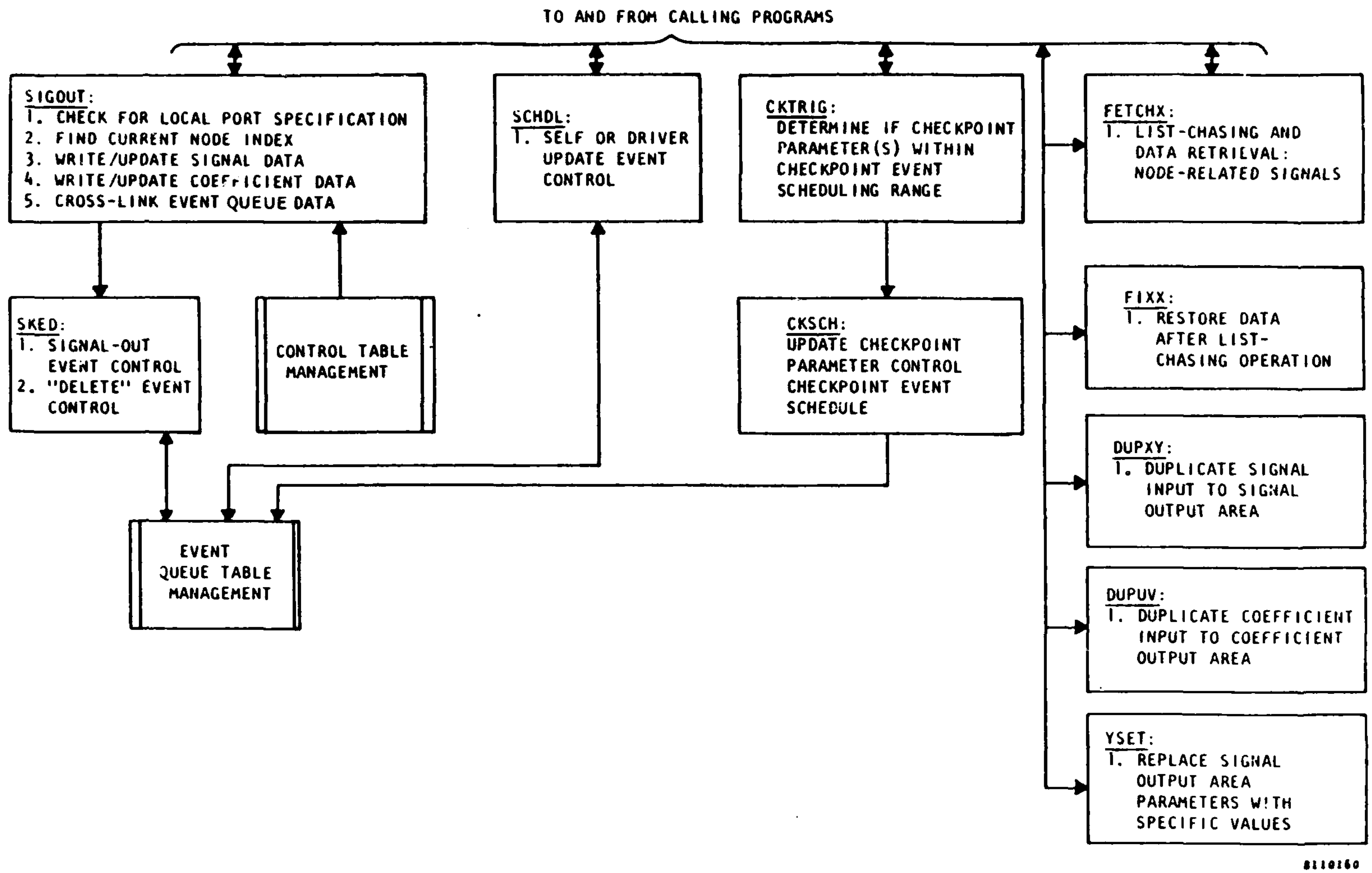


Figure 3-28. Required Internal Utilities

The eight internal tables are interrelated as in the diagram of figure 3-29. This figure depicts the inter-table linkage scheme established within the TSM program. It also depicts the relationship between the Event Queue table and the Control tables.

3.3.3.2.1 Event Queue Table. The form of the Event Queue table is shown in figure 3-19. This table is addressable by "slot" number. The table address number is implicit in the ordering of the table. Each address contains four "components" (columns QA, QB, QC, and QD of the figure). The QC column contains numbers representing any of three kinds of quantity, namely, values of SIMTIME, EVENT CODE, or NODE CODE. The Queue is "threaded" by pointers contained in columns QA and QB. Each "thread" strings like quantities together, as defined by the QC column. For example, a "thread" contains only values of SIMTIME or only values of NODE CODE. More than one of each kind of thread may exist. Each such thread starts with a value of zero stored in the QA column. Succeeding "beads" on the thread are pointed to by the component value in column QB of the same address. For example, in figure 3-19, addresses 1, 4, 7, 9, and 11 all contain a zero in column QA, signifying the head of, or starting point for, five different threads. The thread starting at address 1 contains only values of SIMTIME, ie, it threads through addresses 1, 2, 3, and 101. Similarly, addresses 4, 5, 6, and 102 thread through values of EVENT CODE only.

Threads formed in the manner described are also linked together by the values stored in column QD. Column QD contains values pointing to addresses that establish three links. The links so formed associate a value of SIMTIME with EVENT or EVENTS and with NODE and NODES. For example, in figure 3-19 the QD column component in address 1 contains the value 4. This points to address 4. The QD component in address 4 contains the value 7, which points to address 7.

Link elements form a "chain" comprising a SIMTIME value, and an EVENT CODE value, and a NODE CODE value. The last element (ie, the link terminator) resides in the QD column and points to the Signal record in the SW file, which is associated with the SIMTIME, EVENT CODE, NODE CODE triplet on that link.

The Event Queue Management Processor establishes and maintains the internal contents of the Event Queue table.

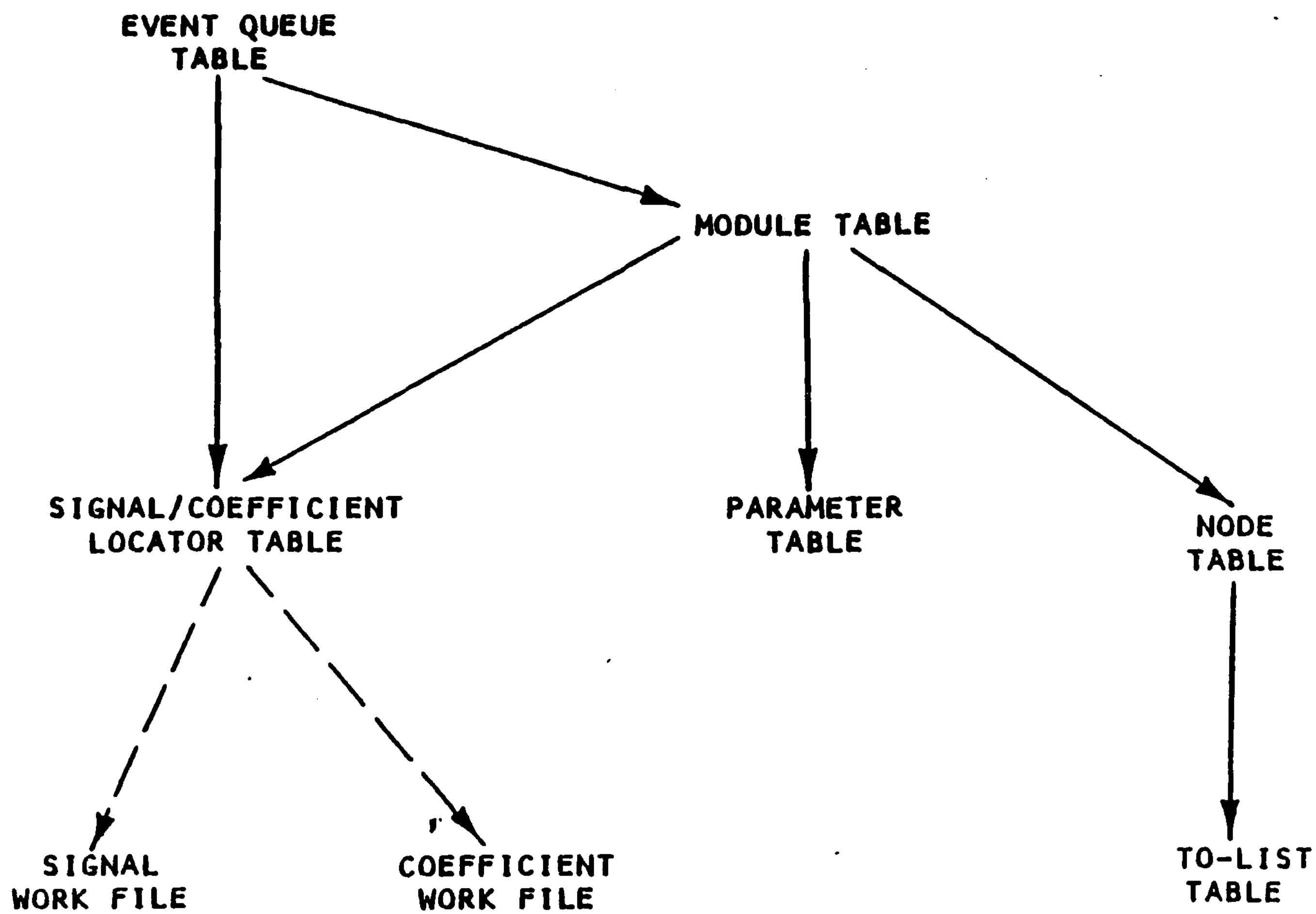


Figure 3-29. Internal Table Interrelationships

The TSM Program, Control/Executive portion maintains a table pointer (INITA) that points to the next event-time associated with the current event being processed from the Event Queue. The INITA pointer provides means to "chase" the "chain" of the internal control table in the order: Event Time - Event Code - Node Code - Node Code Pointer. The chase provides the data needed to hand control to the Process Module portion to effect the simulation processing dictated by the event retrieved and the node involved.

3.3.3.2.2 <u>Simulation Control Tables</u>. The Simulation Control tables (excluding the Event Queue table) control the internal operation of the TSM. These tables are implemented as in figures 3-20 through 3-25. These figures describe the nature of the various table entries, and the interrelationships and intra-relationships among them. These figures also contain specific entry values that depict the intended states of the tables with respect to a specific example shown in the block diagram of figure 3-30.

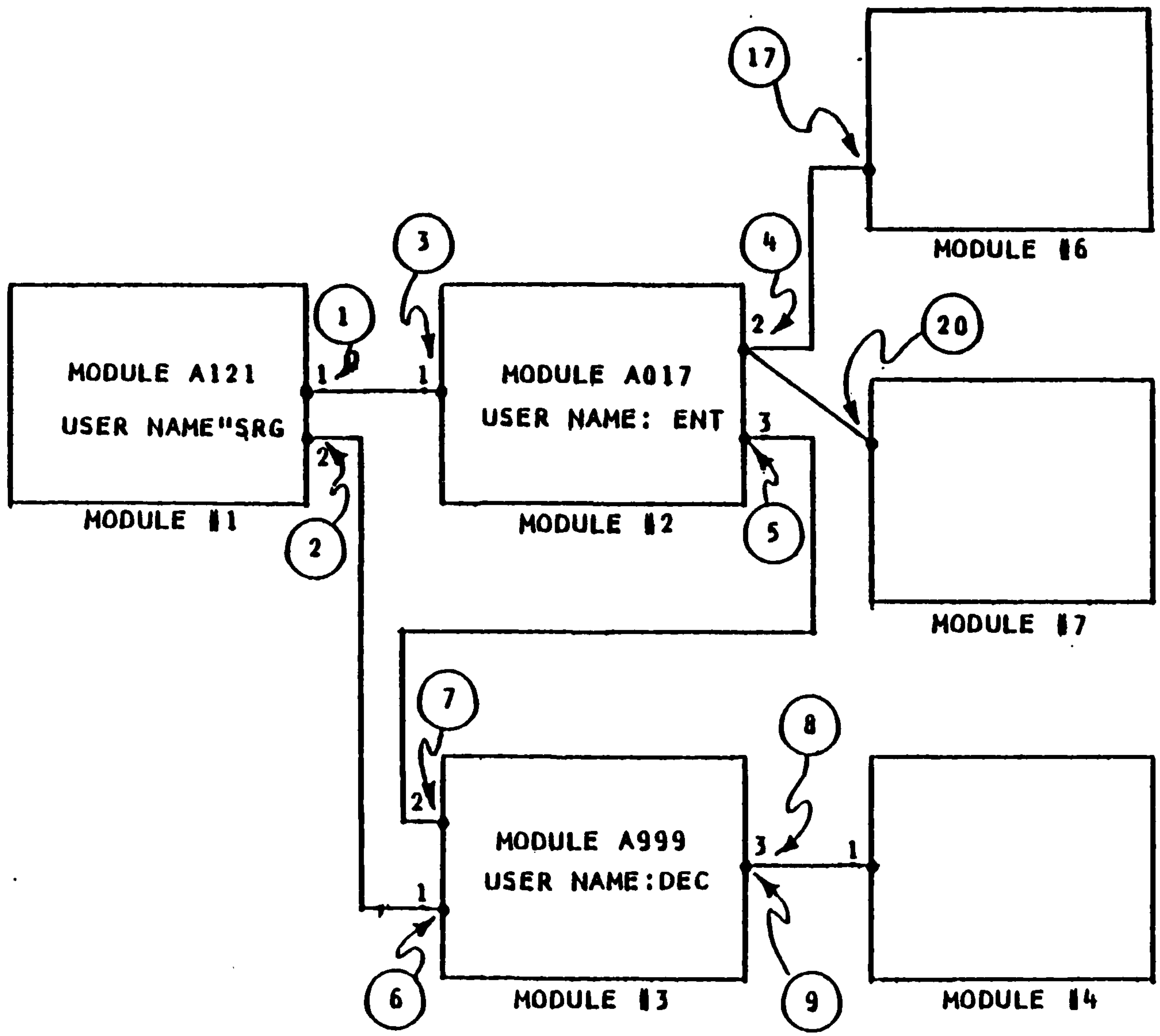
The Control Table Management Processor establishes and maintains the internal contents of the Simulation Control Tables.

3.3.3.2.3 <u>TSM Program Event Codes and Signal List Elements</u>. The Event Codes that appear in column QC of the Event Queue Table are defined in table 3-3.

The Event Codes are listed there in the order: highest priority event type first (ie, Event Code 1), lowest priority event type last (ie, Event Code 8).

The Signal List Record is a collection of descriptors applicable to application modules. Each execution of an application module is accompanied by an updated Signal List Record for that execution. However, cases of an empty or "null" Signal List can arise (ie, cases where all results of execution of a particular application module are reflected in the associated Coefficient Record only).

The fields (elements) of a Signal List Record are defined in table 3-4.



NUMBERS WITHIN CIRCLES ARE NODE-CODE VALUES
ASSIGNED IN ILLUSTRATIVE
EXAMPLE

Figure 3-30. Model Used for Illustrative Examples, Block Diagram

Table 3-3. Event Codes

Event Code	Event Code Value	Meaning
ION	1	An application module of the TSM must be turned on or has been turned on.
IUP	2	An application module output signal is available to update the SW file.
IOFF	3	An application module of the TSM must now be turned off or has been turned off.
ICON	4	An application module control signal has changed state and requests service.
IDEL	5	Delete a signal record from the SW file.
IEND	6	The simulation termination event initiates a poll of ESK control tables for quiescence and to exit simulation.
ICHK	7	Not implemented (reserved for later use in check-pointing).
ISTOP	8	Not implemented (spare).

Table 3-4. Signal List Record Fields (Sheet 1 of 2)

Field	Meaning
TON, TOFF	SIMTIME values when application module was "energized" and was "de-energized." Such an interval is referred to a "segment."
FZ	Center frequency (MHz)
BW	Signal bandwidth value (MHz)
TDUR	Duration of module activation (ie, segment duration) (s)
WVFM	Carrier waveform name or type code
SIGV	Signal voltage (volts)
TZ	Time of Transmission (s)
TERR	Transmitter clock error (s)
ID	Transmitter ID number
R	Transmitter-receiver horizontal range (nmi)
ELEV	Elevation angle relative to receiver (degrees)
AZIM	Azimuth angle relative to receiver (degrees)
PROP	Propagation factor
AMPL	Signal amplitude or signal power value (volts or watts)
PHAZ	Signal phase value (degrees)
DPLR	Doppler rate or value (knots)
NCOF	Number of samples in Coefficient record
SCAL	Scale factor for Coefficient record

Table 3-4. Signal List Record Fields (Sheet 2 of 2)

Field	Meaning
SPAC	Code for Coefficient basis-space
DELT	Incrementing value of time between Coefficient samples (s)
NODE	Node Code for associated output port
NMBR	(as required)
TOK	T_{OK} for module SIMTIME
ENCD	(as required)
TSTOP	TSM execution termination time in SIMTIME
REAL1	(as required)
REAL2	(as required)
INT1	(as required)
INT2	(as required)
INT3	(as required)

3.3.4 Description of the Post-Processor Selector (PPS) Program

The PPS program provides a means to specify the post-simulation signal-data reduction and the processing to be done on the data obtained from the execution of an ICSSM TSM. The PPS program, upon initiation, prompts the user to provide the specifications needed for post-processing actions. After the prompt/response session is completed, the PPS program writes the required specification data to the Post-Processor List (PPL) file and terminates.

3.3.4.1 Major Operations Performed in the PPS Program

A macro-level flowchart reflecting the operations performed in the PPS program is provided in figure 3-31.

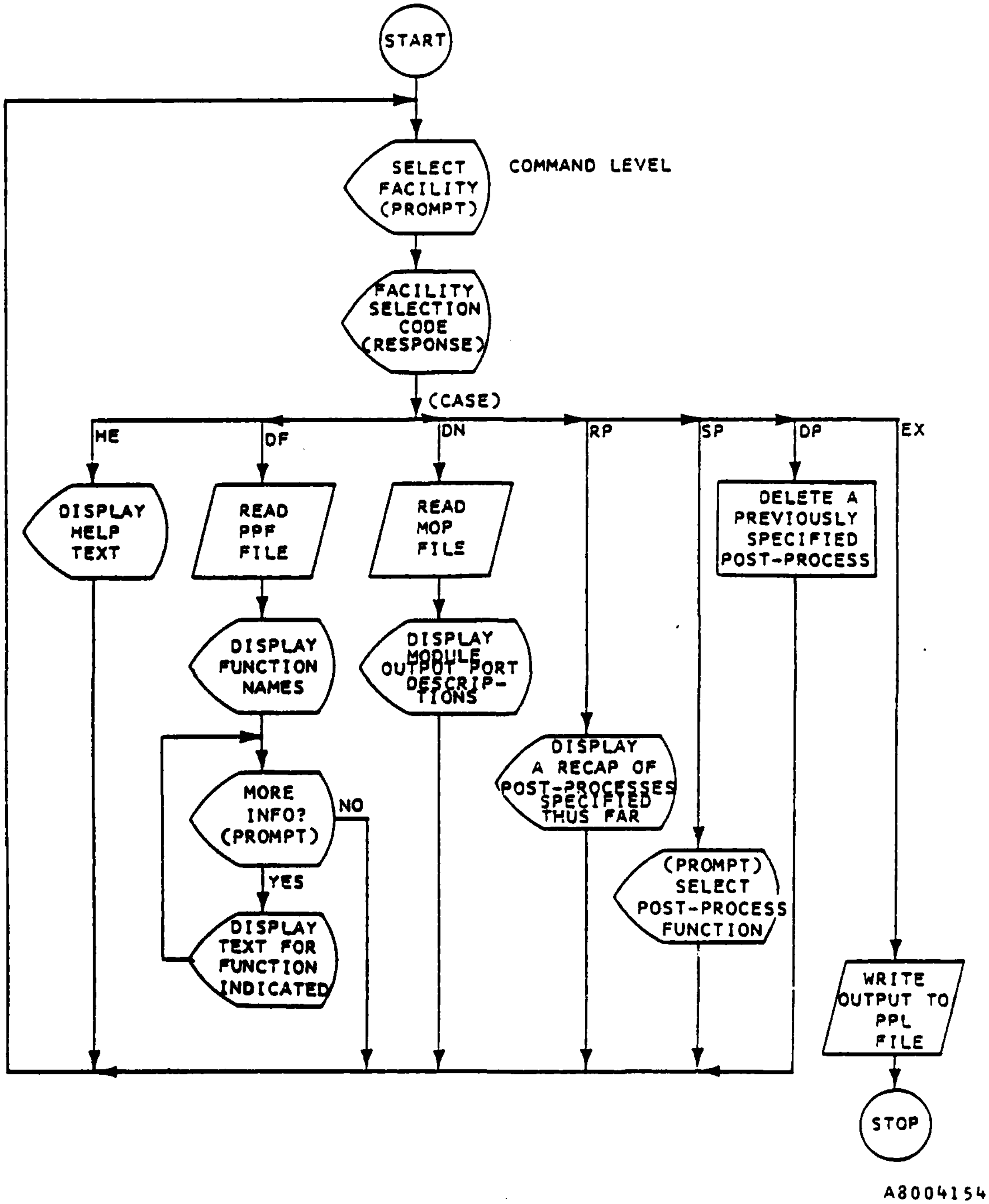


Figure 3-31. Macro-Level Flowchart for Post-Processor Selector (PPS) Program

Hierarchic organization of the PPS program is depicted in figure 3-32.

The PPS program:

- o Allows the user to select from among the post-processing functions available within the PPF file of the LD subsystem.

- o Accepts user selection of the port-output-signals of the TSM to which the selected post-processing functions are applied.

- o Prepares a data file for input to the PPE program, which directs the execution of the specified post-simulation processing operations.

3.3.5 Description of the Post-Processor Exercisor (PPE) Program

The PPE program operates according to the macro-level flowchart of figure 3-33.

PPE program operation terminates automatically, under normal conditions.

3.3.5.1 Major Operations Performed in the PPE Program

The organizational hierarchy for the PPE program is shown in figure 3-34.

1.0 initt -- initialize the plot-10 graphic package.

2.0 term -- inform plot-10 package that the terminal is a tektronics 4014.

3.0 chrsiz -- set the terminal character size (plot-10).

4.0 page -- clear and segment the screen into two parts.

4.1 newpag -- clear the screen (plot-10).

4.2 linhgt -- return the number of raster units per line (plot-10).

4.3 movabs -- move the graphic cursor (plot-10).

4.4 drawabs -- draw a line with the graphic cursor (plot-10).

4.5 anmode -- flush the graphic buffer and return crt to alphanumeric mode.

5.0 nput -- handle a request for the output of a character string to the crt, clearing the screen when needed.

5.1 page -- (see 4.0)

5.2 output -- position the cursor and write a character string to the crt.

6.0 he -- (help) output to the crt a text segment describing the use of this program.

6.1 page -- (see 4.0)

6.2 nput -- (see 5.0)

7.0 lf -- (list functions) output to the crt a listing of the available post-processing functions.

7.1 page -- (see 4.0)

7.2 nput -- (see 5.0)

7.3 pftxt -- output to the crt descriptions of the post-processing functions.

7.3.1 nput -- (see 5.0)

Figure 3-32. PPS Program Hierarchy (Sheet 1 of 2)

8.0 ln -- (list nodes) output to the crt a list of the available output ports. This list is read from a file produced by the precompiler.

8.1 page -- (see 4.0)

8.2 nput -- (see 5.0)

9.0 lp -- (list processes) output to the crt the specifications for processing to be done by the post processor.

9.1 page -- (see 4.0)

9.2 nput -- (see 5.0)

9.3 nfind -- locate a particular section of text in a text array.

9.4 concat -- copy one character string onto the end of another.

10.0 sp -- (specify a process) interactively specify a post-processing function.

10.1 page -- (see 4.0)

10.2 nput -- (see 5.0)

10.3 gap -- interactively specify the ports at which the function is to be applied.

10.3.1 nput -- (see 5.0)

10.4 gparam -- interactively specify the function parameters.

10.4.1 nfind -- (see 9.3)

10.4.2 concat -- (see 9.4)

10.4.3 nput -- (see 5.0)

11.0 dp -- (delete a process) delete one of the specified post processes to be performed.

11.1 nput -- (see 5.0)

12.0 ex -- (exit) write the specified processes into a file to be read by PPE program.

13.0 finitt -- terminate the plot-10 processing.

Figure 3-32. PPS Program Hierarchy (Sheet 2 of 2)

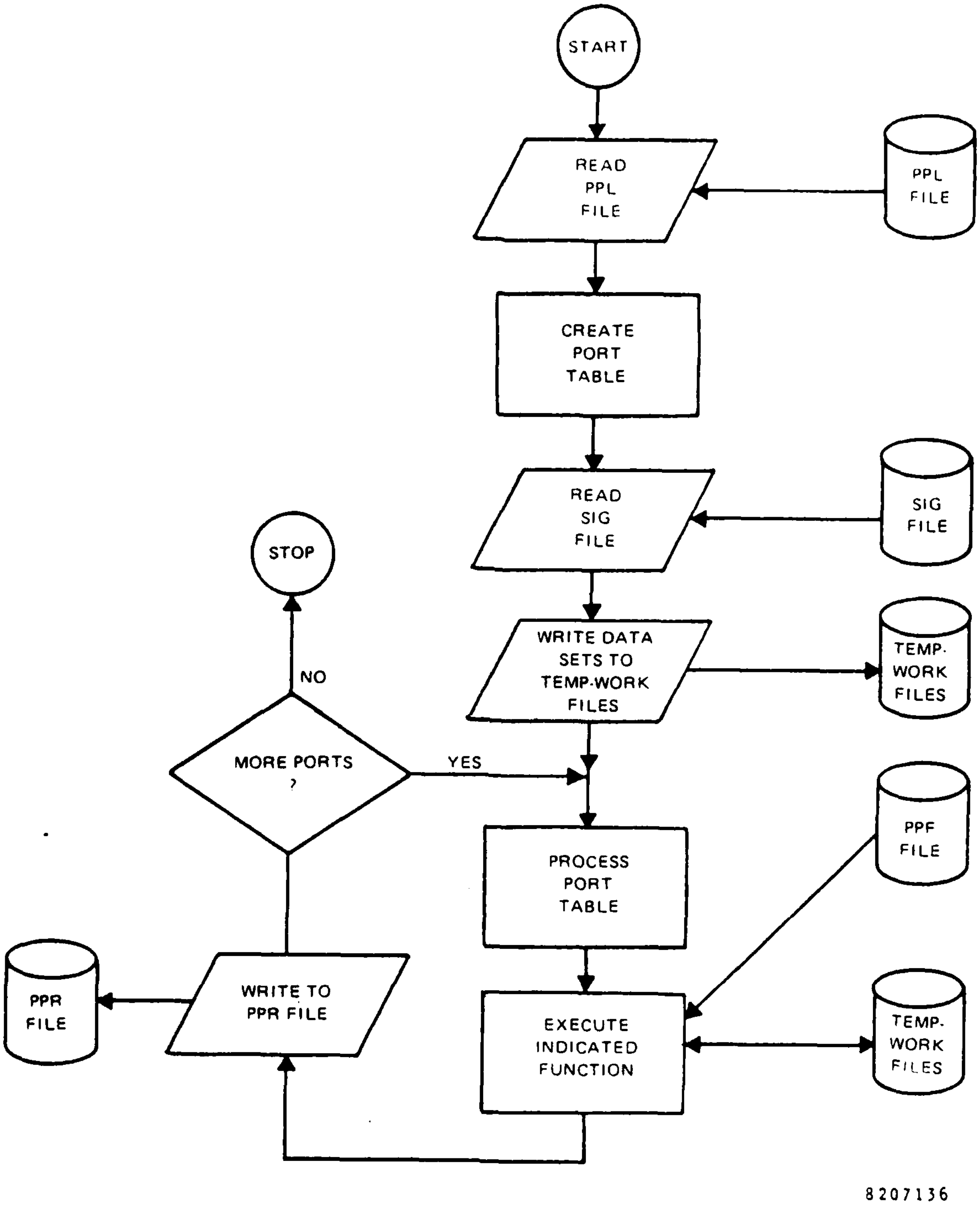


Figure 3-33. Macro-Level Flowchart of Post-Processor Exercisor (PPE) Program

1.0 mklist -- go through the specified processes and construct a list of all the application ports.

  1.1 psort -- sort the application ports by module number and then by port number. Duplicates are eliminated.

2.0 mkfls -- go through the file exer4_dat and copy the required records into the temporary files. One file for each application port.

3.0 exec -- call the appropriate function with its corresponding temporary files and parameters.

  3.1 dofft -- read the temporary file and set up a complex arra- to be passed to cfft2.

    3.1.1 cfft2 -- do a forward or reverse Fast Fourier Transform on a complex array.

    3.1.2 gout -- output the result of the FFT in tabular form.

  3.2 dober -- read the temporary files and determine the bit error rate and inter-error statistics.

    3.2.1 berinit -- initialize the variables used by dober.

    3.2.2 mino -- return the smaller of two integer arguments.

    3.2.3 inc -- increment a 2-integer counter.

    3.2.4 conv -- convert a 2-integer counter into a real.

Figure 3-34. PPE Program Hierarchy

3.4 DETAILS OF ICSSM APPLICATIONS LIBRARY COMPONENT (ALC) IMPLEMENTATION

The following paragraphs describe the programs, subroutines, and files comprising the ICSSM Applications Library Component. The programs described are:

- o Elements comprising the LDAG and the LDUG
- o Programs for Applications Library Maintenance.

3.4.1 Implementation of the ICSSM ALC Directory and Library Elements

The ICSSM Library/Directory contains the files depicted in figure 3-7.

The Library/Directory is comprised of eight data sets organized into an Applications Group (LDAG) and a Utilities Group (LDUG). The LDAG provides the structure for on-line access to the Applications Modules contained in the ICSSM Applications Library.

The functional elements (Modules and DMS) required for communications system modeling reside in the Library Module (LM) file. This LM file is comprised of ASCII files containing all subroutines that can be employed by the user in configuring the TSM.

The LM file contents are divided into "Chapters" (maximum of 20 Chapters). The number of Chapters depends upon the taxonomic structure used in classifying the Modules. The taxonomy is reflected in the contents of the Library Chapter (LC) file and the Library Chapter Detail (LD) file, in that the LC file contains data on the most general classification of library modules, and the LD file contains data describing those library modules comprising each of the general classes (or Chapters) delineated in the LC file. The structure of these two files is analogous to chapter headings and within-chapter details provided by the "Table of Contents" of most textbooks.

3.4.1.1 Directory Structure Implementation in LD Applications Group

The LD Applications Group (LDAG) is organized into three ASCII files that contain interrelated data (see figure 3-35).

- o File LC is a sequential file that contains one 80-position ASCII record for each Chapter in the LD subsystem. Each record contains an 18-character Chapter description field and a 3-digit field containing the number of Applications Library modules in that Chapter. The remainder of each record is blank.

- o File LD is a direct-access file containing 80-position ASCII records. Each record contains a 6-character module name field (the library-assigned name for the module). The remainder of the record is blank. There may be up to 20 such records per Chapter record in the LC file.

- o File Module Description and Help (MDH) is a random-access file containing 80-position ASCII records. Each module record in the LD file generates a group of 58 records in the MDH file. Each such group is organized as in table 3-5.

3.4.1.2 Description of Library Module (LM) File Entries

The LM file is a user-defined file in the ICSSM host computer applications or user file area. It contains either FORTRAN or compiled (ie, relocatable) versions of each Applications Library entry, or both, depending upon user needs, upon ICSSM Librarian requirements, or upon host computer requirements. The LM file contains three types of entries:

- o Those used as general-purpose modeling elements via the MC subsystem (ie, in the TSM).

- o Those intended as general-purpose test modules for ICSSM maintenance purposes.

- o Those intended for ICSSM validation tests (these may have general-purpose modeling application as well).

All types of entries described above are referenced by the LD, LC, and MDH files.

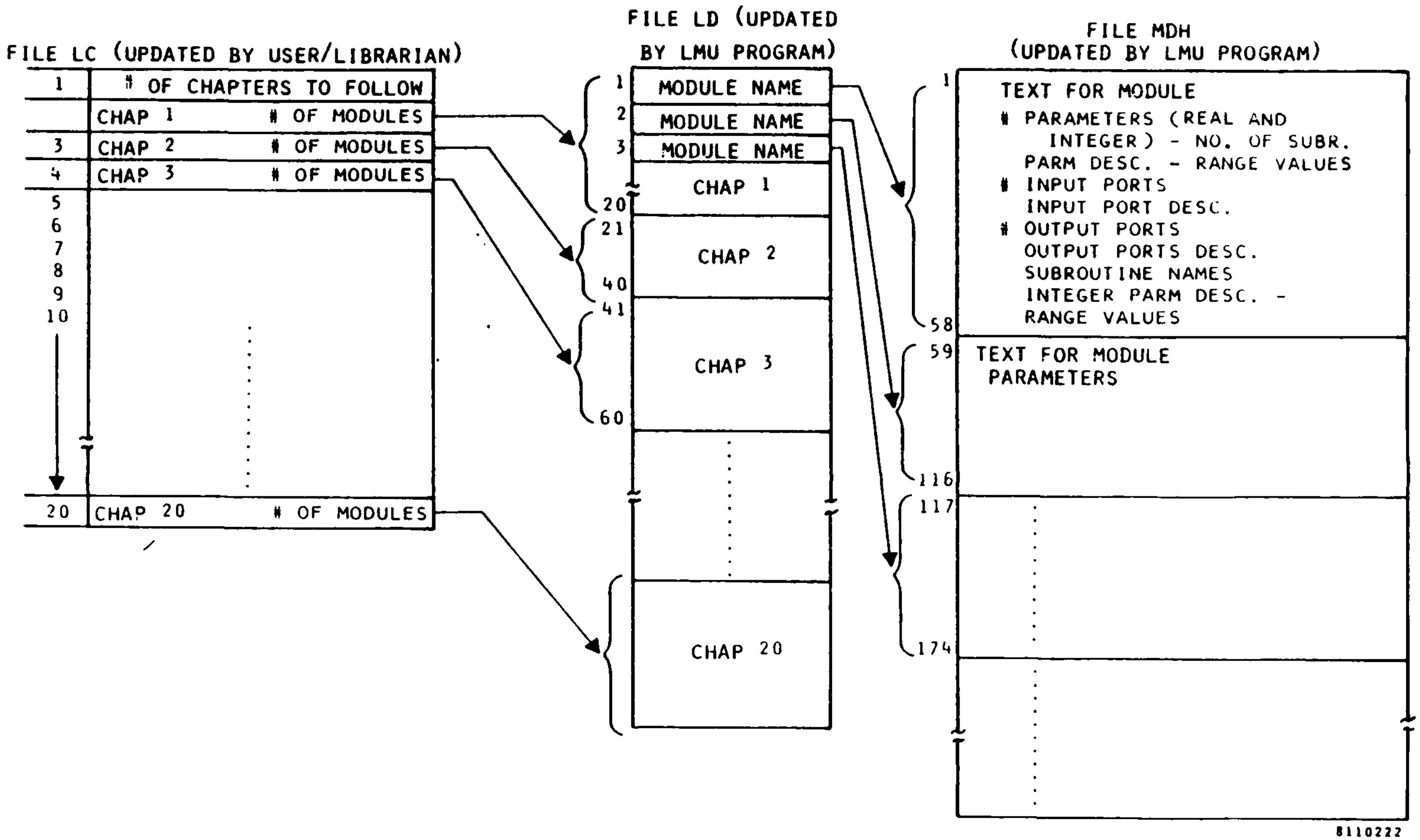


Figure 3-35. Interrelationship of Applications Library Directory Files

Table 3-5. MDH File Data Items

Position No. of Record within a Group	Description of Group Record Contents
1, 2, 3	Text describing nature of module and method of use
4	Description of "hidden" parameters
5	Number of parameters required by module, and number of subroutines required by module
6 thru 25	Descriptions and names of floating-point parameters for module
26	Number of input ports defined for module
27 thru 31	Descriptions of input ports of module and instructions for their use
32	Number of output ports defined for module
33 thru 37	Description of output ports of module instructions for their use
38	List of subroutine names CALLed by module
39 thru 58	Descriptions and names of integer parameters for module

3.4.1.3 Description of General Purpose LM File Items

The general-purpose items (modules) in the LM file are listed in tables 3-6 and 3-7. (Refer to paragraph 3.3.2.1)

Table 3-6. Applications Modules (Class 1 Items) in the LM File (Sheet 1 of 2)

Module Name	Module Function
TSGP00	A source of SFSK-modulated binary data.
CCHP02	A model of a communication channel with additive band-limited Gaussian noise.
RMFP02	Model of a synchronized SFSK waveform-matched filter.
RDCP00	SFSK decoder/decision model.
TCPP01	A bit stream comparator/bit error computing module for use in tests and validations.
TSGP01	A source of voice-emulation waveforms.
TCPP02	A power spectrum difference comparator for use in tests and validations.
TNCP00	A QPSK bit-stream encoder.
CAPP00	An all-pass propagation channel model with additive correlated white Gaussian noise.
RPLP00	A phase-locking loop model with correlated noise in the continuous wave reference signal.
RDMP01	A demodulator of QPSK-modulated continuous wave that includes the correlated noise due to synchronization errors in phase-locking loop.
RDCP01	A QPSK decoder model.
GDLP00	A charge coupled device (ccd) delay line model.
RMFP00	A model of a SAW filter matched to a SFSK chip waveform.

Table 3-6. Applications Modules (Class 1 Items) in the LM File (Sheet 2 of 2)

Module Name	Module Function
RMFP01	A model of a SAW filter matched to a selectable PN code.
GLPP00 GLPP01	Lowpass elliptic filter models.
GBPP00 GBPP01	Bandpass Butterworth filter models.
GHPP00	A highpass filter model.
GAMP00	A model of a switched summing amplifier.
GAMP01	A model of an amplifier with automatic gain control.
GLMP00	A model of an amplifier limiter.
REDP00	A model of an envelope detector.
GCPP00	A model of an integrator comparator.
GSTP00	Finds the first eight moments in of an input signal.
GSTP01	Finds bit error rate, mean of the error free runs, and the standard deviation.

Table 3-7. Dependent Modeling Routines (Class 2 Items) in the LM File (Sheet 1 of 3)

Subroutine Name	Subroutine Function
XMTR (FZP, BWP, WVFMP, SIGVP, TERRP)	A routine for initializing channel parameters.
DPSKBT (SEED, OLDBIT, NENBIT, PICBIT)	A binary DPSK modulation source.
SFSK (T, R, V, TREF, FMHZ, TERR, TOD, SI, SQ)	A generator of sinusoidal frequency/ phase-shift keyed continuous wave signal.
KODE (TX, TPD, AI, AQ)	A generator of in-phase and quadrature samples for the 'analytic signal.'
NOISE (V, N, SIGMA, ISEED)	A source of white Gaussian noise.
ERROR (MODN)	A routine for writing error messages to the EXJ.
INITB	An initializing routine used by module COMPA.
BFWRI	A routine used to transfer data within the TSM.
AM (AMOD, V, N)	A continuous wave amplitude modulator.
CNOIS (SIGMA, ISEEDX, ISEEDY, TDECOR, XN, YN)	A source of time-correlated additive white Gaussian noise voltage.
UNQPSK (KS, KC, A, B)	A QPSK demodulator routine.
INTGRT (M, NCOPF)	A general-purpose integrator routine.
CCVM (AR, AI, BR, IB, CR, CI, N)	Routine to multiply vector B by complex A.
WEIGHT (WI, WQ, NEL, G, TCONST, KOUNT, GT, ALPHA, TOT)	A routine to compute weight coefficients in adaptive antenna model.

Table 3-7. Dependent Modeling Routines (Class 2 Items) in the LM File (Sheet 2 of 3)

Subroutine Name	Subroutine Function
CUTVM (AR, IA, BR, BI, IB, CR, CI, N)	Routine to perform complex vector multiplication.
ALPAS	Routine for signal manipulation in all-pass propagation channel model.
GEOM (XTT, ITX, TNOW, R, PHI, S, THD, SDOT, SG, THG, SGDOT)	A routine to compute geometry of receiver signal effects based on transmitter/receiver antenna relationships.
POSIT (X, T)	Routine to update transmitter antenna position in doppler calculations.
DIRECT (R, TH, PG, SPRD, DT, VT, VR)	Routine to compute angles theta and phi of transmitter relative to receiver.
XANT (TH, PH)	Routine to compute flat-earth range.
RBLAD (TH, PH)	Routine to compute antenna element spacing in antenna array.
SPEC (R, S, TH, PH, SPRD, DT, VR, PSL, RHO, PSI)	Routine to compute effects of specular reflection in propagation channel model.
GREFL (SN, CN, F, AMAG, FAZE)	Routine to compute effects of ground reflection in propagation channel model.
ROUGH (SINTH, ALMBDA, RUFF, HT, HR, RH)	Routine to calculate roughness factor for ground reflection modeling.
FLUCT (RHOP, PHBP, EP, RHOEFP, PHBEFP)	Routine to model fluctuation of ground specular reflection multipath.
STORE (INDX, PR, SQ, THQ, PHI, SPRDQ, DTG, VRC, RHO, PSI, S, TH, PH, SPRD, DT, VT, VR, XT, TNOWP, THD, SDOT, SGDOT)	Modeling utility for storing module parameters into a working array.

Table 3-7. Dependent Modeling Routines (Class 2 Items) in the LM File (Sheet 3 of 3)

Subroutine Name	Subroutine Function
REFSFS (NRFBTS, REFAMP, IERC, REFU, SAVQTS, INSAMP, DELTCP, SAVSD, FMHZP, IPCPAT)	Generates SFSK-modulated reference modulation signal.
UNWIND (ARRAY, NCOFX, BARRAY, NELEM)	Routine to convert floated version of binary number to bit pattern of ones and zeroes.
CRRSMP (FMHB, CPHAS, NSAMP, TX, IERC, SEPOCH, CARRI, CARRQ)	Generates sample of in-phase and in-quadrature carrier.
CHIP (DELTC, KDWVFM, NSAMP, IERC, SMPCI, SMPCQ, SEPOCH)	Generates samples SFSK chip-modulation waveform.
MDULAT (IT, TQ, NSAMP, IERC, SQTS, CARRI, CARRQ, SMPCI, SMPCQ)	Generates SFSK-DPSK modulated carrier.
MDULAR (TI, TQ, NSAMP, IERC, SQTS, CARRI, CARRQ, SMPCI, SMPCQ)	Generates sampled bandpass correlation reference.
RESTOR (IDXP, RP, SQ, THQ, PHI, SPRDQ, DTG, VTG, BRG, RHO, PSI, S, TH, PH, SPRD, DT, VR, XT, TNOWP, THD, DCOT, SGDOT)	Modeling utility for storing working parameter values in parameter array.
RPIGRG (AMPI, N, M)	Normal random variable generator (zero-mean, unit-variance).

3.4.1.4 Description of Support Utilities (SU) File Data Items

The LD Utilities Group (LDUG) is organized into two files that contain FORTRAN subroutines used in supporting applications within the ICSSM system. The SU file contains the AP-120B Array Processor Emulator routines, and certain other routines used in several subsystems of the ICSSM system and in the design of Applications Library modules. These are defined in tables 3-8 and 3-9.

Table 3-8. AP-120B Emulator Routines (Class 3 Items) in the SU File (Sheet 1 of 2)

Subroutine Name	Subroutine Function
VCLR (C, N, N)	Clears a vector array (CLEARS C).
VMOV (A, I, C, K, N)	Moves a vector array (A C).
VNEG (A, I, C, K, N)	Negates all vector components (-A C).
VADD (A, I, B, J, C, K, N)	Adds two vector arrays (A + B C).
VSUB (A, I, B, J, C, K, N)	Subtracts two vecotr arrays (A - B C).
VSMULT (A, I, S, C, K, N)	Multiplies vector by scalar (S x A C).
VDOTPR (A, I, B, J, C, N)	Dot product of two vectors (A . B C).
VLDG (S, I, C, K, N)	Base 10 log of a vector (LOG_{10} A C).
VEXP (A, I, C, K, N)	Exponentiate a vector (EXP(A) C).
VSIN (A, I, C, K, N)	Sine of vector (SIN(A) C).
CVMUL (A, I, B, J, C, K, N, DUMMY)	Complex vector multiply (A/B C).
VCOS (A, I, C, K, N)	Cosine of a vector (COS(A) C).
VATAN (A, I, C, K, N)	Arctan of a vector (ARCTAN(A) C).

Table 3-8. AP-120B Emulator Routines (Class 3 Items) in the SU File (Sheet 2 of 2)

Subroutine Name	Subroutine Function
VMUL (A, I, B, J, K, N)	Multiple two vectors (A x B C).
VDIV (A, I, B, J, C, K, N)	Divide a vector by a vector (A/B C).
VSQRT (A, I, C, K, N)	Square root of a vector (A C).
SVE (A, I, C, N)	Adds vector components ($a_1 + a_2 + \ldots a_1$ C).
VFLT (J1, I, C, K, N)	Converts integer vector components to floating point representation (FLOAT (A) C).
POLAR (A, I, C, K, N)	Rectangular to polar conversion of vector.
RECT (A, I, C, K, N)	Polar to rectangular conversion of vector.
CVMAGS (A, I, C, K, N)	Complex vector magnitude squared (A^2 C).
MTRANS (A, I, C, K, NRC, NCC)	Transpose a matrix (A^T C).
MMUL (A, I, B, J, C, K, NRC, NCC, NCA)	Multiply two matrices (A x B C).
CONV (A, I, B, J, C, K, N, M)	Convolve A with B (A * B C).
CFFT2 (C, N, IF)	Fourier transform of array (F(C) C').
VABS (A, I, B, J)	Absolute value of vector (ABS(A) B).
MAXV (A, F)	Component of A having maximum value.
BITRV3 (C, N)	Rearranges coefficients obtained in AP-120B subroutine CFFT2.
APPUT (A, I, N, IDUMMY)	Stores array A in array processor buffer.
APGET (A, I, N, IDUMMY)	Retrieves array A from array processor buffer.

Table 3-9. Simulator Internal Support Routines (Class 4 Items) in the SU File (Sheet 1 of 2)

Subroutine Name	Subroutine Function
YSET (TIME, TDURP, DELTP, IDP, NMBRP, NCOFP, SPACP)	Sets up operating control values for on-coming TSM module.
SIGOUT (TIME, NOUT, NEW, KNTRL)	Transfers active TSM module output to EK COMMON area and creates output Event Queue entry.
INIT	Initializes pointers in Event Queue and control tables of TSM.
SCHDL (NOTE, TIME)	Updates Event Queue table to schedule new event for subsequent TSM processing.
EHAND (ISIG)	Regulates passage of control between EK Mainline and active TSM module.
HANDL	Loads and stores data between active TSM module and EK COMMON.
SLOAD (ISIG)	Transfer Signal List data from SW file record to EK main storage area.
DUPUV	Transfer data from UBLOCK to VBLOCK in EK COMMON area.
DUPXY	Transfer data from XBLOCK to YBLOCK in EK COMMON area.
FETCHX (NODE, INDX)	Chases time/event/node chain for Signal List pointer.
FIXX	Restores original contents of TSM XBLOCK COMMON after thread chase.
SKED (TIME, ISIG, LEVNT, JTO, KTO)	Updates/schedules events for self-driven or externally driven TSM modules.
CFILE	Closes all TSM files and terminates TSM operation.

Table 3-9. Simulator Internal Support Routines (Class 4 Items) in the SU File (Sheet 2 of 2)

Subroutine Name	Subroutine Function
DSTOR (M, R, INDX, INITL, ID, IREC, IFN, IDUP, IRTN, NDIM)	Dynamic storage management routine: links and unlinks beads on threads of threaded main storage array; maintains linkages in TSM Management Control Tables.
LOCATE (TBL, R, INITL, TID, IT, ISTAT, ND)	Chases linkage chains in TSM Control Tables.
MAKE (TBL, R, INDX, INITL, TID, NEW, IT, IER, ND)	Inserts new thread bead onto thread of threaded array. Inserts new links into TSM Management Control Tables.
BREAK (TBL, INDX, INITL, IT, ND)	Deletes bead from thread of threaded main storage array. Deletes links in TSM Management Control Table.
ENSRT (TIME, EVNT, RNODE, ISIG, IREC, JREC, KREC)	Inserts new properly positioned events in TSM Event Queue Table.
FERR1 (N)	Outputs Control Table or Event Queue Table error condition message to EMM file and terminates TSM execution.

3.4.2 Description of ICSSM Applications Library Maintenance Program

The ICSSM system Applications Library Maintenance/Update (MU) Subsystem is comprised of a single program - the LMU program. The structure of the MU Subsystem is shown in figure 3-36.

The MU subsystem provides facilities that allow the ICSSM Librarian/Custodian to modify the contents of the files comprising the LDAG.

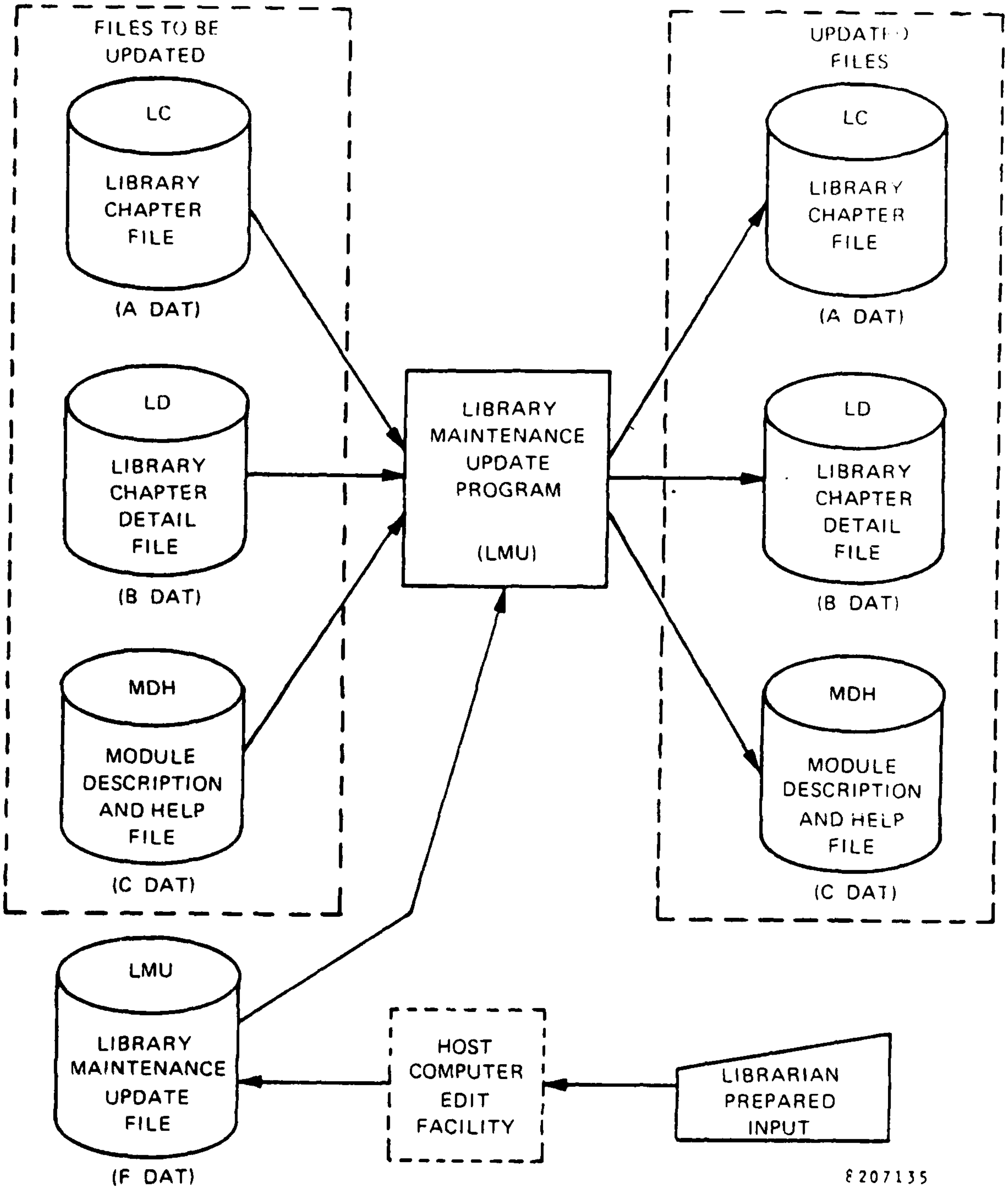


Figure 3-36. Structure of Maintenance/Update (MU) Subsystem

The MU subsystem employs a Librarian/Custodian-prepared updated (LMU) file. Execution of the Library Maintenance/Update (LMU) utility program results in appropriate modifications of the contents of the files of the LDAG. Actual insertion of the Applications Module into the LM file is done via the host computer public-file management system.

The LMU program operation:

- o Changes the "Number of Modules" field in the appropriate Chapter record of the LC file
- o Adds Chapter detail records into the LD file to reflect Applications Library (ie, LC file and MDH file) additions
- o Adds the 58 records needed to register an applications module into the MDH file.

The inputs to the LMU program are the LC file, the LD file, the MDH file, and the LMU file. The LMU file is a sequential ASCII file. It is prepared by ICSSM host computer edit and file-build utilities. Descriptions of the Data Items for the LMU file are provided in table 3-10.

Table 3-10. Formats for LMU File Record Data Items

Item Name	Item Description	Item Format
N	Number of Modules	I3
MODN	Module Name	3A2
CHAPR	Chapter Reference Number	I3
NPAR	Number of Parameters	I2
NHIDP	Number of Hidden Parameters	I4
MTEXT	Module Description Text (3 Records)	3(80A1)
NUMSBR	Number of Subroutines Used by Module	I2
PDES	Parameter Descriptions	15A2
PIODES	Input Port, Outport Port Descriptors	9A2
NUMIP	Number of Input Ports	I1
NUMOS	Number of Output Ports	I1

The output of the LMU program contains updated versions of the LC, LD, and MDH files.

SECTION IV

VALIDATION TEST RESULTS

4.1 INTRODUCTION

The ICSSM system performed successfully under functional tests executed according to the procedures recounted in the Test Analysis Report (ref 54). In addition, a simulation model of a broad-band communication receiver synchronization processor of current interest was implemented with Applications Library modules designed for the purpose. The results of verification tests of this model are reported in detail in the Test Analysis Report.

4.2 RESULTS

The results reported and analyzed here pertain to test No. 2 described in figure 3 on page 40 of volume II of the Test Analysis Report of the ICSSM SYSTEM APPLICATIONS LIBRARY MODULES DEVELOPMENT PROGRAM. The input signal was produced by the TACOM/ICSSM Bridge Program (see Appendix D), which was adjusted to deliver noise only. The noise parameter, FNOISE, measured relative to the Chip Matched Filter input was then equivalent to the receiver input noise factor multiplied by the square of the voltage gain prior to the Chip Matched Filter. The TACOM Bridge settings are listed in table 4-1. The simulated communication gain factors that were used are listed in table 4-2.

4.3 ANALYSIS AND COMPARISON OF TEST RESULTS

The theoretical values of the statistical moments have been calculated at the outputs of the TACOM/ICSSM Bridge, the AGC'd amplifier and the Envelope Detector and then compared with the experimental test results.

4.3.1 TACOM/ICSSM Bridge Program Output Calculation

The output of the TACOM/ICSSM Bridge is intended to simulate a band-limited white gaussian noise process. The usual parameters of such a noise process are:

B = 14.56 MHz
R = 50
F = 10,000
kT = 4e-21

The definitions of the parameters B, R, f and kt are as follows:

B = The baseband noise bandwidth (one half of the radio frequency bandwidth).

R = Receiver radiation resistance.

F = Receiver noise factor relative to the input of the receiver. In this case it is equal to the product of the noise factor relative to the antenna and the available power gain between the antenna and the input to the receiver.

kT = The product of Boltzman's constant (k = 1.38e-23) and the noise temperature (T = 290).

Table 4.1 TACOM/ICSSM Bridge Program Parameter settings

CSIG1	0.0
CSIG2	0.0
FNOISE	10,000.0
Noise Bandwidth	14.56 MHz

The definitions of CSIG1, CSIG2, and FNOISE are as follows:

CSIG1 = Multiplying factor applied to the direct path signal component generated by the TACOM system.

CSIG2 = Multiplying factor applied to the multipath signal components (excluding the direct component) generated by the TACOM system.

FNOISE = receiver noise factor relative to the input of the simulated receiver. In this case, FNOISE is equal to the product of the noise factor relative to the antenna and the available power gain between the antenna and the input to the simulation.

Table 4-2. Simulated Communication System Gain Factors

Chip matched filter insertion gain		1.0
PN matched filter insertion gain		1.0
10 MHz BPF insertion gain		1.0
Limiter insertion gain		0.9225
20 MHz BPF insertion gain		1.0
Envelope detector insertion gain		0.5807
6 MHz LPF insertion gain		0.5
Summing amplifier:	Input #1 gain factor	2.739
	Input #2 gain factor	2.0
100 kHz HPF insertion gain		1.0
CCD delay line insertion gain		1.0
10 MHz LPF insertion gain		0.5
Integrator-comparator:	amplifier gain factor	-20.0
	integrator factor	-0.05
	plus comparator input factor	0.50
	minus comparator input factor	0.03

The block diagram of the model (figure 4-1) indicates the connection of Applications Library modules for the validation test.

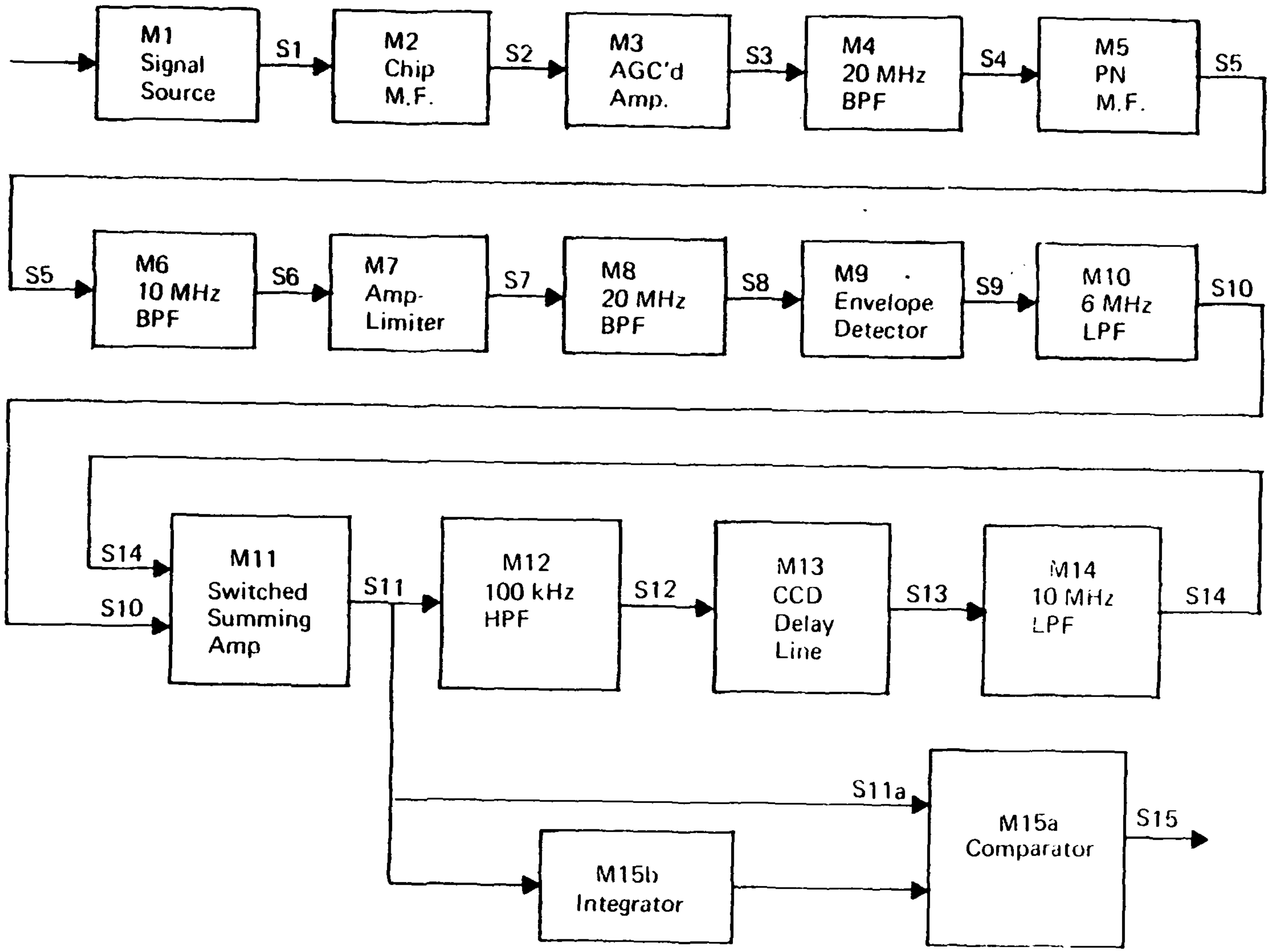


Figure 4-1. Simulation Model Configuration Block Diagram

The corresponding first and second moments of the in-phase and quadrature-phase signal components are:

m1 = 0.0
m2 = R*kT*B*F* = 0.2912e-07

Table 4.3 This table lists the ICSSM test results calculated by the Post Processor and the corresponding theoretical results for the following cases:

(a) Output of the TACOM/ICSSM Bridge.
(b) Output of the AGC'd Amplifier.
(c) Input to the Envelope Detector.
(d) Output of the Envelope Detector.

	Test Results	Theoretical Values
(a) Output of TACOM/ICSSM Bridge		
ml, In-phase.	-0.6625e-05	0.0
m2, In-phase.	0.2751e-07	0.2912e-07
ml, Quadphase	0.9859e-06	0.0
m2, Quadphase	0.2904e-07	0.2912e-07
(b) Output of AGC'd amplifier		
ml, In-phase.	-0.2585e-01	0.0
m2, In-phase.	0.1000e+00	0.1019e+00
ml, Quadphase	0.2778e-02	0.0
m2, Quadphase	0.1059e+00	0.1019e+00
(c) Input to Envelope Detector		
ml, In-phase.	0.1517e-02	0.0
m2, In-phase.	0.1767e-03	0.1677e-03
ml, Quadphase	0.4840e-03	0.0
m2, Quadphase	0.1587e-03	0.1677e-03
(d) Output of Envelope Detector		
ml	0.9373e-02	0.9426e-02
m2	0.1120e-03	0.1131e-03

4.3.2 AGC'd Amplifier Output Calculations

The TACOM/ICSSM Bridge noise factor, F, was selected to produce an input to the AGC'd amplifier that is sufficiently strong to cause its operation to be within the AGC control range. Consequently, the mean envelope amplitude should be 0.4. The corresponding first and second moments of the in-phase and quadrature-phase signal components are:

m1 = 0.0
m2 = (2/PI)*(0.4**2) = 0.1019

The test results are compared with the theoretical values in item (b) of table 4-3.

4.3.3 envelope detector Input/Output Calculations

In this test, the theoretical output moments are calculated on the basis of the experimentally determined input second moment, which is estimated to be equal to the average of the input in-phase and quadrature-phase second moments. Therefore,

input moments

m1 = 0.0
m2 = (0.1767e-03 + 0.1587e-03)/2 = 0.1677e-03

output moments

m1 = G*SQRT (PI*m2/2) = 0.9426e-02
m2 = (G**2) *2*m2 = 0.1131e-03

where G is the effective insertion gain of the Envelope Detector.

The test results are compared with the theoretical values in items (c) and (d) of table 4-3.

That the two distributions are quite close is indicated by the observations that the difference between their average is 0.014 and the RMS value of the differences between them is 0.038. This closeness indicates that the simulation is very accurate with high probability. This is a reasonable conclusion because the comparisons of theory with simulation results must take into consideration the statistical verity illustrated by the following analogy: 50 "heads" out of 100 tosses of a particular coin does not absolutely prove that the coin is perfectly balanced but it does indicate that the probability that the coin is balanced is very high.

4.3.4 The test results given are the sample mean, m1, and the sample variance, m2, calculated by the Post Processor from 1820 consecutive sample values from the simulation. The theoretical results are the corresponding ensemble means and variances of the prototype noise signal amplitudes. The test results are average values calculated from the results of a simulation, whereas, the theoretical values are the corresponding ensemble averages obtained from the simulation prototype. There are, therefore, two distinctions between the test results and the theoretical:

1. The test results are based on the results of a simulation and the theoretical results on the prototype of the simulation;

2. The test results are time averages and the theoretical are ensemble averages.

Each of these distinctions will contribute to the discrepancies between corresponding test results and theoretical values.

The first distinction will yield the same discrepancies as are found in ordinary deterministic simulation studies. These discrepancies are computational errors caused by the approximations necessary in formulating a time-discrete, amplitude-discrete simulation model corresponding to a time-continuous, amplitude-continuous prototype.

The second distinction will yield the same errors that always occur when estimating the statistical properties of a stochastic process by measuring the corresponding time averages which are also random variables. In particular, consider the ergodic stochastic process (x(t)) with mean value

m = E(x)

and variance

v = E(SQR(x - m)).

The sample mean, m1, and sample variance, m2, are both random variables and have the following central moments (where N is the number of sample values used to calculate the sample mean and sample variance):

```
mean (m1) = E(m1)
          = m

var (m1) = E(SQR (m1-E(m1)))
         =v/N

mean (m2) = E(m2)
          = (N-1)*v/N

var (m2) = E(SQR(m2-E(m2)))
         =(N-1)*((N-1)*E((x-m)**4) - (N-3)*SQR(v))/(N**3)
```

The "normalized error" can also be used for comparing the test results and theoretical results. The normalized error is the ratio of the total discrepancy to the standard deviation of the test result standard deviation. In particular, the normalized error of m1 is

```
e(m1) = (m1 - mean (m1))/(SQRT(var(m1)))
```

and the normalized error of m2 is

```
e(m2) = (m2 - mean (m2))/(SQRT(var(m2)))
```

The following table contains the values of these normalized errors for the data listed in table 4-3.

Signal Point	E(m1)	E(m2)
In-phase TACOM/ICSSM Bridge Output	-1.66	1.65
Quad-phase TACOM/ICSSM Bridge Output	0.246	-0.622
In-phase AGC'd Amp Output	-1.57	-0.219
Quad-phase AGC'd Amp Output	0.168	0.575
In-phase Env. Detector Input	2.27	0.772
Quad-phase Env. Detector Input	0.724	-0.699
Envelope Detector Output	-0.0965	-0.194

It is only the error component due to the first distinction that can be considered an inaccuracy in the simulation since the second distinction is not related to the simulation process. Unfortunately, it is not possible to determine precisely what part of the total error is due to the second distinction since e(m1) and e(m2) are also random variables.

The effects of this randomness can be reduced by comparing the distribution of normalized errors with the Gaussian distribution function since the two curves will converge as the number of normalized error values increases (and also the number of sample points increases if the noise signal in non-Gaussian) given that the overall simulation is accurate.

These distributions are graphed in figure 4-2.

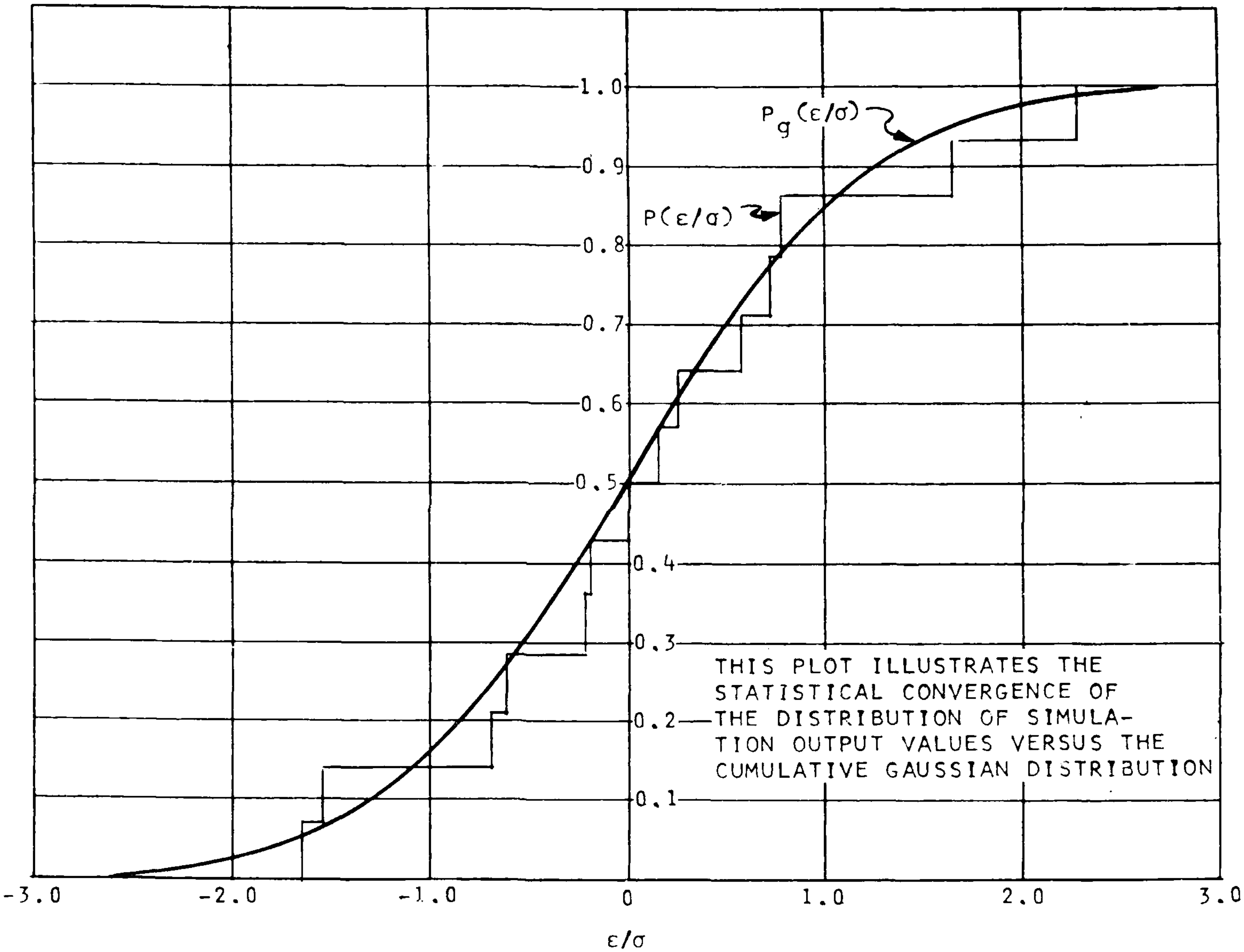


Figure 4-2. The Gaussian Distribution versus the Distribution of the Normalized Errors.

SECTION V

RECOMMENDATIONS AND CONCLUSIONS

5.1 INTRODUCTION

The results reported on and discussed in Section IV and general experience with the operation of the ICSSM system, prompt the recommendations outlined below. These recommendations fall into two categories, pertaining to: operations of the ICSSM facility, and enhancements and improvement to the ICSSM structure. A conclusion is also included.

5.2 RECOMMENDATIONS PERTAINING TO ICSSM OPERATION

Recommendations that address the efficiency of the ICSSM system in actual use are discussed in this paragraph.

5.2.1 Simulation Timing Measurements

The complex interrelationships between the need for flexibility and generality in ICSSM, on the one hand, and the need for speed and ease-of-use, on the other hand, suggest that ICSSM simulation executions be studied using an in-core trace/timing software utility. This will provide a profile of where (what portions of computer code) the typical simulation spends most of its time. This would be useful to determine where improvements and "tightening" of code would have the greatest impact on speed and simulation efficiency. It would also ensure that more arbitrarily selected sites for coding improvement would not exact a penalty in reduced generality and flexibility without a justification in terms of trade-off with speed and efficiency.

5.2.2 Overhead Reduction in Simulation

The Applications Library module designer has a strong interest in design for general use. The re-entrant philosophy for ICSSM Applications Library module design helps in this direction. However, the desires to generalize and to isolate communications - theoretic "unit" processes in individual modules - results in the tendency to design and use many separate modules in configuring a TSM. Each module used in a TSM represents an individual passage of control and data to and from the simulator executive (ie, the Exercisor Kernel) each time that that module is invoked during a simulation. This passage of control and data represents an "overhead" from the user's point of view.

This suggests that a study and analysis be made to determine the minimum overhead possible, to devise rules of trade-off for designing modules and TSM that approach this minimum, and to devise a means to determine the "optimum size" for Applications Library module design.

One possibility for good trade-off is to design a means whereby individual modules could be gathered into a "unit" that could then be used as a simulation element in its own right. This would allow individual modules to be designed, manifesting generic or canonical modeling functions, and yet be combined into a "supermodule" or "unit" that interfaces with the simulation executive. Thus, preconfiguration of such "units" before setup and simulation of a "large" model could be a feasible means of finding and implementing the correct trade-off point.

5.2.3 Speed Improvement by Use of Array Processor

The ICSSM system design, in part, presumes the presence of an array processor peripheral to the host computer. Emulator routines in the ICSSM Applications Library provide the means to design modules using array processor "functions" or "calls" even in the absence of an array processor. Efficient use of the array processor requires the transfer of relatively large data vectors and arrays of data to and from that processor. Requirements on module design and the algorithmic details of particular models constructed for ICSSM simulations, on the other hand, may dictate processing of relatively small vectors and arrays. Under these conditions, an array processor would not significantly increase the speed of some modules or processors. The overall benefit of an array processor to an ICSSM simulation depends on the effectiveness of applying array manipulations to modeling algorithms. This suggests that a study and analysis to find optimum rules for using array processor "calls" be pursued. These rules, if formulated, would determine the type and location within modules where array processor manipulation would be optimally effective.

5.3 RECOMMENDATIONS PERTAINING TO IMPROVEMENTS IN ICSSM STRUCTURE

Recommendations that address ease-of-use, adaptability, expandability and applicability of the ICSSM system are discussed in this paragraph.

5.3.1 Recursive Modeling

When a complete simulation model (TSM) has been configured and completed, the resultant software assemblage is treated as an application program by the host computer operating system, indistinguishable from any other application program. From the ICSSM point of view, if a TSM could be treated as a subroutine structured according to ICSSM requirements, it itself could be installed in the Applications Library and used in a more complete ICSSM TSM as if it were a module. In this way, the ICSSM system could bootstrap itself into more and more complicated modeling structures, thus providing a kind of recursive modeling capability. While there are some practical problems involved (eg, ICSSM TSM read from and write to particular files, and this I/O programming is embedded in the structure of the Exercisor Kernel, so that difficult file management problems would arise if steps were not taken to mitigate them), this possibility suggests that a study and design to produce a bootstrap facility be pursued.

5.3.2 Quick-Look Capabilities

Simulations can take a long time to run to completion. Simulation models are relatively difficult to design and configure even with the aid of a system like ICSSM. Uncertainty always exists at the outset of a simulator execution as to whether the model is valid. A means should be provided within ICSSM to make a quick appraisal of the simulator output data (and all intermediate data as well) to determine if the model is executing correctly before large amounts of computer resources are invested. Such a "quick-look" facility would be a worthwhile enhancement to the ICSSM system.

5.4 CONCLUSION

Experience with the ICSSM shows that it is a viable, worthwhile, powerful means of providing simulation-analytic insights and solutions/results for difficult communications system design and analysis problems.

Further development of ICSSM Applications Library modules, particularly modules that are generic models of communications system elements, is strongly indicated. Practical use of the ICSSM system will perfect it as a convenient and relevant tool for the simulation and analysis of communications systems.

APPENDIX A

REFERENCES

1 Moss, R. W., Hammond, J. L., Donaldson, E. E., "Interactive Communication Systems Modeling Study," Interim Technical Report, RADC TR-76-155, May 1976.

2 Moss, R. W., Rice, R. W., Sentz, D. R., "Interactive Communication Systems Modeling Study," Final Technical Report RADC TR-77-42, February 1977, ADA037833.

3 Losson, Thomas P., McRae, Daniel D., "Broadband Digital Modem," Rome Air Development Center, Final Technical Report, RADC TR-76-117, May 1977, ADA025399.

5 Gilbert, E. N., "Capacity of a Burst-Noise Channel," BSTJ, vol. 39, p. 1253, 1960.

6 Elliott, E. O., "A Model of the Switched Telephone Network for Data Communications," BSTJ, vol. 44, pp. 89-110, 1965.

7 Berger, J. M., and Mandelbrot, B., "A New Model for Error Clustering in Telephone Circuits," IBM J. Res. & Dev., vol. 7, p. 224, 1963.

8 Sussman, S. M., "Analysis of the Pareto Model for Error Statistics on Telephone Circuits," IEEE Trans. on Communication Systems, col. CS-11, p. 213, 1963.

9 Prabhu, V. K., "Error Rate Considerations for Coherent Phase-Shift Keyed Systems With Co-channel Interference," Bell Syst. Tech. J., vol. 48, no. 3, pp. 743-767, March 1969.

10 Rosenbaum, A. S., "Binary PSK Error Probabilities With Multiple Co-channel Interferences," IEEE Trans. Commun. Tech., vol. COM-18, pp. 241-253, June 1970.

11 Goldman, J., "Statistical Properties of a Sum of Sinusoids and Gaussian Noise and its Generalization to higher Dimensions," Bell Syst. Tech. J., vol. 53, no. 4, pp. 557-580, April 1974.

12 Fang, R., and Shimbo, O., "Unified Analysis of a Class of Digital Systems in Additive Noise and Interference," IEEE Trans. Commun., vol. COM-21, pp. 1075-1091, October 1973.

13 Benedetto, S., Giglieri, E., and Castellani, V., "Combined Effects of Intersymbol, Interchannel, and Co-channel Interference in M-ary CPSK Systems," IEEE Trans. Commun., vol. COM-21, pp. 997-1008, September 1973.

14 Jeruchim, M. C., and Lilley, F. E., "Spacing Limitations of Geostationary Satellites Using Multilevel PSK Signal," IEEE Trans. Commun., vol. COM-20, October 1972.

15 Rosenbaum, A. S., "PSK Error Performance Gaussian Noise and Interference," Bell Syst. Tech. J., vol. 48, no. 2, pp. 413-442, February 1969.

16 Prabhu, V. K., "Error Probability Upper Bound for Coherently Detected PSK Signals with Co-channel Interference," Electron., Lett., vol. 5, no. 16, pp. 383-386, August 1969.

17 Aein, J. M., "On the Effects of Undesired Signal Interference to a Coherent Digital Carrier," Inst. for Defense Analyses, Paper P-812, February 1972.

18 Aein, J. M., and Turner, R. D., "Effect of Co-channel Interference on CPSK Carriers," IEEE Trans. Commun., vol. COM-21, pp. 783-790, July 1973.

19 Rosenbaum, A. S., and Glave, F. E., "An Error Probability Upper Bound for Coherent Phase-Shift Keying with Peak Limited Interference," IEEE Trans. Commun., vol. COM-22, pp. 6-16, January 1974.

20 Glave, F. E., and Rosenbaum, A. S., "An Upper Bound Analysis, for Coherent Phase-Shift Keying with Cochannel, Adjacent-Channel, and Intersymbol Interference," IEEE Trans. Commun., vol. COM-23, pp. 586-597, June 1975.

21 Krishnamurthy, J., "Bounds on Probability of Error Due to Co-channel Interference," IEEE Trans. Aerosp. Electron. Syst., vol. AES-11, pp. 1373, November 1975.

22 Wilmut, M. J., and Campbell, L. L., "Signal Detection in the Presence of Co-channel Interference and Noise," IEEE Trans. Commun., vol. COM-20, pp. 1153-1158, December 1972.

23 Goldman, J., "Detection in the Presence of Spherically Symmetric Random Vectors," IEEE Trans, Inform. Theory, vol. IT-22, pp. 52-59, January 19 6.

24 Miller, G. A., and Lickleder, J. C. R., "The Intelligibility of Interrupted Speech," p. 167, JASA, vol. 22, 1950.

25 Kryter, K. D., "Methods for the Calculation and Use of the Articulation Index," p. 1695, JASA, vol. 34, November 1962.

26 Gumbel, E. J., "Statistics of Extremes," New York Columbia Univ., Press 1958.

27 Weinstein, S. B., "Theory and Application of Some Classical and Generalized Asymptotic Distributions of Extreme Values," IEEE Trans. Inform. Theory, vol. IT-19, pp. 148-154, March 1973.

28 Moss, R. W., Rice, R. W., Sentz, D. R., "Interactive Communication Systems Modeling Study," Rome Air Development Center, RADC-TR-42, Feb. 1977, ADA037833.

29 Shannon, Robert E., "Simulation Modeling and Methodology," Bicentennial Winter Simulation Conference Proceedings, Vol 1, pp 9-15, December 1976.

30 RADC Computer Software Development Specification, Specification No. CP 0787796100B. Code Ident 07877, Rome Air Development Center, N.Y., 28 February 1977.

31 Maynard, Denis R., "Introduction to the RADC R&D Computer Facility," Rome Air Development Center, RADC-TR-77-61, March 1977, ADA038657.

32 McEvoy, J. B., Sturdevant, N. J., "RADC's Digital Communications Experimental Facility," AFCEA, Signal Magazine, Vol XXVIII, No. 10, July 1974, pp. 4-12.

33 Moss, R. W., Rice, R. W., Leong, P. K., "Model for Interactive Design and Analysis of Communication Systems," Bicentennial Winter Simulation Conference Proceedings, Vol 1, pp. 185-189, December 1976.

34 Clema, J., Scarpino, F., "A General Purpose Tool for Interactive Simulations" Bicentennial Winter Simulation Conference Proceedings, Vol 2, 475-484, December 1976.

35 "Digital Transmission System Design," DCEC TR-3-74.

36 "DCS Digital Transmission System Performance," DCEC TR-12-76.

39 Bello, Philip A., et al, "Line-of-Sight Techniques Investigation," Rome Air Development Center, Final Report, RADC-TR-74-330, January 1975, ADA006104.

40 "Adaptive Modulation and Error Control Techniques," IBM Corp, Rome Air Development Center Final Technical RADC-TR-66-169, May 1966, AD484188.

41 Schmandt, Frederick D., "DCA Systems Evaluation - Phase 1," Rome Air Development Center, In-house Report, RADC-TR-76-358, February 1977, ADA038607.

42 Losson, Thomas R., McRae, Daniel D., "Broadband Digital Modem," Rome Air Development Center, Final Technical Report, RADC TR-76-117, May 1977, ADA025399.

43 Jones, Norman G., et al, "Microwave Data Transmission Test Program - QPSK Modems," Rome Air Development Center, Final Report, RADC TR-74-274, November 1974, ADA002840.

47 Colby, G. V., et al, (C) "A Concept for Jam Resistant Secure Voice Communications,"(U), Lincoln Laboratory Project Report, TST-4, April 1976.

48 Beusch, J. U., et al, (C) "A Jam Resistant Secure Voice Communications System Concept,"(U), Project Report TST 7, ESD TR 76-349, 14 December 1976, ADC008875.

49 Beusch, J. U., et al, "Aircraft Nulling Concepts for Jam Resistant Secure Voice Communications," Project Report, ESD TR-76-377, 13 January 1977.

50 "Functional Description for Interactive Communication System Simulation Model (ICSSM) Extension,"(U), Hazeltine Corp. Report #6437R1 July 21, 1981.

51 "System Specification for Interactive Communication System Simulation Model (ICSSM) Extension," Hazeltine Corp. Report #6382R1 September 1981.

52 "Programming Specifications for Interactive Communication System Simulation Model (ICSSM) Extension," Hazeltine Corp. Report #6387R1 October 1981.

53 "User Manual for Interactive Communication Systems Simulation Model (ICSSM) Extension," Hazeltine Corp. Report #6390R1 February 1982.

54 "Test Analysis Report on Interactive Communication System Simulation Model,"(U), Hazeltine Corp. Report #6477.

55 "Program Maintenance Manual for Interactive Communication System Simulation Modeling (ICSSM) System," Hazeltine Corp. Report #6397R1.

APPENDIX B

LIST OF ABBREVIATIONS AND ACRONYMS

The abbreviations and acronyms used in this document are listed below. This list does not include abbreviations and acronyms that are in accordance with MIL-STD-12.

ALC	Applications Library Component
AWGN	Additive White Gaussian Noise
CCS	Coefficient Checkpoint Status
CDMA	Code Division Multiple Access
CE	Control Executive
CTC	FORTRAN Checkpoint Trigger Controller
CTM	Control Table Management
CW	Coefficient Work
DCLF	Systems Design and Analysis Section
DCS	Defense Communication System
DICEF	Digital Communications Experimental Facility
DMS	Dependent Module Subroutines
EK	Exercisor Kernel
EMM	Error Message and Module
EQP	Event Queue Table Processor
ES	Exercisor/Simulator
ESP	Exercisor Simulator Process
EVJ	Event Journal
EXJ	Execution Journal
FCBA	FORTRAN Common Block Alignment
FFT	Fast Fourrier Transfer
FMD	FORTRAN Model Description
FTH	Frequency-Time Hopped
FTR	Final Technical Report
GDU	Graphics and Display Utilities
ICSSM	Interactive Communication System Simulation Model
IMS	Intermediate Model Specification
LC	Library Chapter
LD	Library/Directory; Library Chapter Detail
LDAG	Library/Directory Applications Group
LDUG	Library/Directory Utilities Group
LM	Library Module
LMU	Library Maintenance Update

MC	Model Configurator
MCS	Model Configurator Select
MCP	Model Configurator Precompiler
MDH	Module Description and Help
MME	Minimization of the Mean-Square Estimation Error
MMT	Master Module Table
MOP	Module and Output Port
MSD	Model Specification Description
MSI	Multiple Sinusoid Interference
MSR	Model Specification Retention
MTE	Model Table Extract
MU	Maintenance/Update
NLI	Node List Table
OTC	FORTRAN Output Data Transfer Controller
PLI	Parameter Table
PM	Process Module
PN	Pseudorandom Noise
PP	Post-Processor
PPE	Post-Processor Exercisor
PPF	Post-Processor Function
PPL	Post-Processor List
PPS	Post-Processor Selector
RADC	Rome Air Development Center
RCFS	Relevant Computable Functions
SC	Simulation Component
SCS	Signal List Checkpoint Status
SFC	Set FORTRAN Common
SIG	Signal Output
SIMTIME	Simulated Time
SSL	Scientific Subroutine Library
SU	Support Utilities
SW	Signal Work
TCS	TSM Checkpoint Status
TSM	Target Simulation Model

APPENDIX C
LIST OF TSM ROUTINES AND THEIR FUNCTIONS

BREAK	Deletes entry from EK Control tables.
CFILE	Closes files and stops program execution. This routine employs calls to plot-10 wind-up routines.
CHEKIN	Restores EK common block variables (see p 4.4 of Program maintenance manual vol. 2 part 2) to values read in from the checkpoint status file (TCS) and positions the EXJ, EVJ, SIG, ASCII EVJ, and ASCII SIG FILES (EXER2.DAT, EXER3.DAT, EXER4.DAT, EXER8.DAT, EXER9.DAT) at the record corresponding to the checkpointed status.
CHECKPOINT	Outputs EK common block variables to checkpoint status file (TCS). Copies SW file (EXER6.DAT) into the SCS file (EXER10.DAT) and the CW file (EXER7.DAT) into the CCS file (EXER11.DAT). Outputs checkpoint EPoch record to the EXJ, EVJ, SIG, ASCII EVJ, and ASCII SIG files (EXER2.DAT, EXER3.DAT, EXER4.DAT, EXER8.DAT, EXER9.DAT).
CKSCH	Updates checkpoint control parameter and inserts a checkpoint event into the event queue.
CKTRIG	Determines if checkpoint parameters are within checkpoint range.
CPTIM	Employs a call to MULTICS operating system function: VIRTUAL_CPU_TIME to find elapsed CPU time within an execution.
DATSUP	Determines IF signal record and coefficient record associated with a particular port are to be recorded in output files EXER4.DAT and EXER9.DAT.
DSTOR	Maintains forward and backward pointers in EK control tables.
DTIM	Employs a call to multics operating system subroutine CU-$AF to obtain current date and real time.
DUPUV	Copies the values contained in the common block U BLOCK into common block V BLOCK.
DUPXY	Copies the values contained in the common block X BLOCK into common block YBLOCK.
EHAND	Interprets and controls processing of events according to the particular event code (all events except "IEND").

ENSRT	Inserts time, event and node code triplet into the event queue and records the cross-reference to signal data base in event list membership pointer.
EXER	Mainline routine for ICSSM EK. Controls initialization and input. Fetches events from event queue. Determines: If system sataus data should be updated and displayed; If control should be transferred to routine finish; If control should be transferred to event handling routine EHAND. Updates EK checkpointable variables.
FERR1	Outputs error message whenever pointer overflow in link data base occurs.
FETCHX	Retrieves signal list and coefficient list data from SW and CW files and stores that data in the common blocks XBlock and UBlock.
FINISH	Invoked when "IEND" event code is encountered. Polls modules to ensure that module processing is complete before termination of program execution.
HANDL	Copies data from common MODLST into common blocks PARAM and IPARAM for module processing. Copies data in common blocks PARAM and IPARAM back into common MODLST following module processing.
INIT	Initializes: Event queue and control table pointers; Variables in the common blocks: EVENT, CONST, ST, and CPS; Variable to contain time at which a checkpoint event occurred; Inserts initial "IUP" event and the "IEND" event in the event queue. This routine employs calls to plot-10 start-up routines.
INPUT	Reads input data from MTE (ETBL.DAT) file.
LOCATE	Chases chains and threads in EK control tables.
MAKE	Inserts new entry in any of the EK control tables.
MSCLR	Erases terminal screen, returns cursor to upper left corner of screen, initializes line counters (employs plot-10 calls).
OPFILE	Opens files used in TSM execution.

PROCES	Transfers Control to appropriate applications library module for processing.
PRTHDR	Prints header text at top of terminal screem. (employs plot-10 calls).
READD	Reads text from system status files and outputs that text onto the terminal screen.
SCHDL	Schedules (ie. inserts in event queue) "IUP" events for a self-updating module or a driving module.
SIGOUT	Finds current node index. Writes/updates signal list data into the SW file writes updates coefficient list data into the CW file insert pointers in the signal and coefficient data base pointer tables.
SKED	Schedule new events in the event queue whenever signal output from a module occurs.
SLOAD	Loads signal record from SW file and coefficient record from CW file into EK common blocks XBlock and UBlock.
SSM	Updates system status variables and outputs updated information to system status file.
SSMAPP	Appends lines of text to text already on terminal screen (employs plot-10 calls).
SSMDVR	Drives routines for display of system status monitoring information (employs plot-10 calls).
SSMFWO	Controls text output to the terminal screen (employs plot-10 calls).
SSMINT	Sets the values of display control variables used in system status monitoring output.
TRIGCK	Updates checkpoint control parameter and transfers control to checkpoint operation routine.
TSMT	Calculates percentage of simulation execution completed and estimates the projected number of hours until execution termination.
YSET	Sets signal values in common YBLOCK.
WRITT	Writes text out to specified unit number.

APPENDIX D

TACOM/ICSSM BRIDGE

The TACOM model is used to simulate communications system effectiveness during realistic Air Force scenarios. A scenario is broken into time slices, and for each aircraft at a given time, information about the signal environment is calculated. The information at each "snapshot" of the signal environment is as follows:

(1) Signal power level from an omni-directional antenna.

(2) Signal power level from an adaptive antenna.

(3) Range from a transmitter.

(4) Range rate of the transmitter relative to the receiver.

(5) Carrier frequency of the transmission.

(6) Indication of the desired signal of interest.

The TACOM/ICSSM Bridge generates samples of a lowpass (baseband) representation of signals. Samples of both the in-phase and quadrature portions of these signals are computed and, as described earlier, each portion of the signal is kept separate throughout. All important data needed to comprise the signal is contained in the SIGNAL LIST associated with signals output from the bridge. It is a simple process then to apply the carrier to the baseband signals, thus recovering the modulated signal. It is seen that the TACOM/ISCCM bridge is most easily used when ICSSM target simulation models (TSM) are generated using modules which operate on baseband signals. It is also seen, however, that the bridge may be used for TSM which utilize bandpass signal representations. The signals must first be processed through a modulation module prior to input into the remaining modules of the TSM.

The TACOM/ICSSM bridge may be used in the following two ways:

In the first case, an ICSSM user configures a system in which a transmitter is modeled and generates a "desired" signal. In this case, all signals generated by the TACOM System are considered external to the signal environment and treated as "noise" to the desired signal. ICSSM suitable signals are contained in the bridge file in the format used for ICSSM Model partitioning.

Figure 1 shows the configuration of Case 1 where module 'TRANSMITTER' generates the desired signal. ICSSM suitable signals are contained in the bridge file in the format used for ICSSM partitioning. (See paragraph)

Since in this case ICSSM module 'TRANSMITTER' generates the desired signal, the in-phase and quadrature signals for the first TACOM signals (input signals 1 and 2) are ignored. The in-phase and quadrature portions of the signal sum are now combined into a single signal record in module, 'COMBINE'. Note: Combining does not imply summing. Rather, combining is a process whereby in-phase and quadrature signals are inserted into the same signal record.

Samples of the 'desired' signal are sent out on port #1 and summed with the total signal noise environment from port #16 within module 'CHANNEL'.

The sum of all signals now in the system is sent out on port #4 to the receiver. Thermal noise (WGN) is added to the desired signal and sent to an identical receiver on port #5. Both receiver outputs are compared in 'OTERM'.

Figure 2 is an alternate case 1 in which the "desired" signal is read from a model partitioning file. The model partitioning file contains a signal generated by an ICSSM transmitter on a previous exercisor kernel (EK) execution of a target simulation model (TSM). Case 1 (figure 1 or figure 2) enables study of receiver performance with alternate transmitters in a realistic (TACOM generated) signal environment. However, in these cases the "desired" signal will not have associated with it the effects resulting from propogation (ie, multipath, transmitter/receiver orientation geometry, antennas). Whereas, TACOM generated signals will have undergone all of these effects. As a result, many users will want to use the TACOM/ICSSM bridge in the manner defined in case 2.

In the second case, four signals are input from the "BRIDGE FILE". Ports 1 and 2 contain the in-phase and quadrature desired signals respectively. Added to the desired signal is thermal noise (modeled as white gaussian noise (WGN). Ports 3 and 4 contain the sum of all TACOM generated signals for this 'snapshot' for in-phase and quadrature respectively (WGN included).

The "BRIDGE INPUT DRIVER" module is a self-updating module, which reads equal numbers of samples of each TACOM generated signal residing in the "BRIDGE FILE", during each module update event.

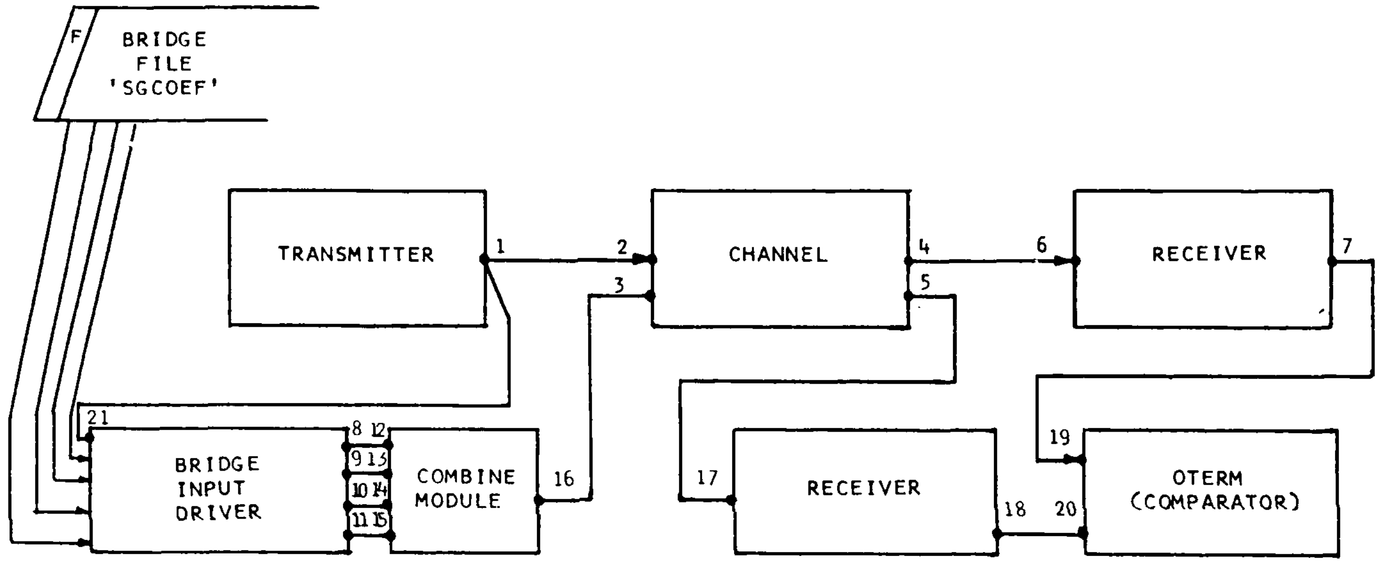


Figure 1. A TSM in which a Model of a Transmitter generates a "desired signal."

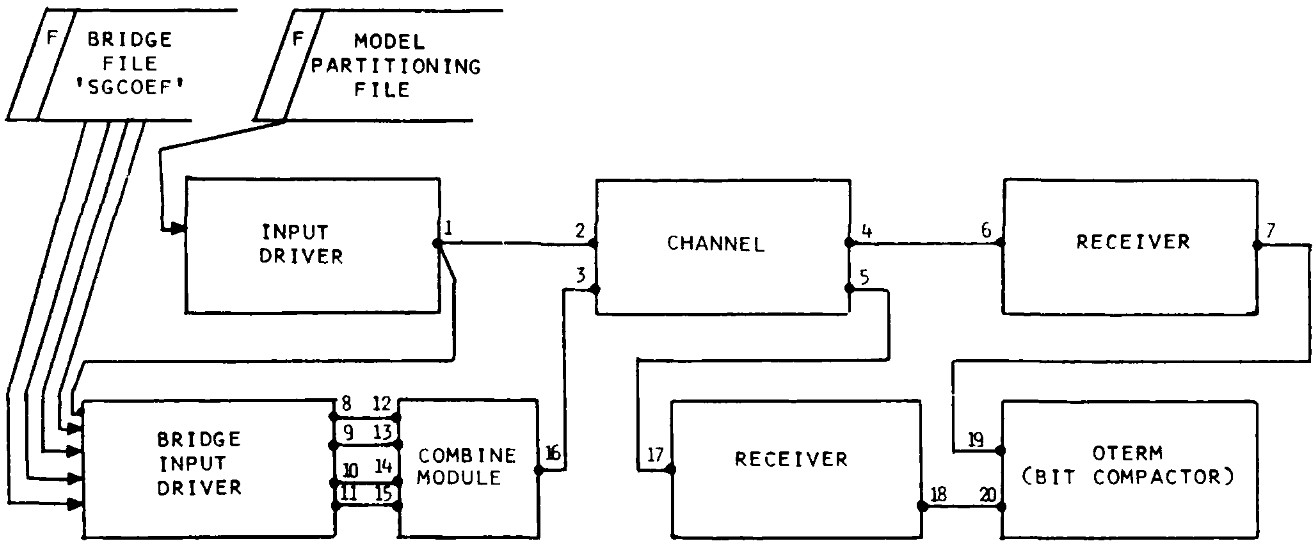


Figure 2. A TSM in which the "desired" signal is read from a model partioning file.

Figure 3 shows the in-phase and quadrature signals kept separate throughout ICSSM target simulation model (TSM) execution. The desired signal plus WGN is input through a receiver on port No. 13 and 14. The sum of all signals plus WGN is input to an identical receiver on port No. 9 and 10. A comparison of both receiver outputs is made in module 'OTERM'.

This use of the TACOM/ICSSM bridge allows for each signal to have undergone the effects of ground multipath, transmitter/receiver orientation, and antenna. This allows for evaluation of receiver performance within a realisitic signal environment on realistic signal samples.

Note that if the user wished, he could have inserted a combining module immediately after the 'BRIDGE INPUT DRIVER' to combine in-phase and quadrature signals for both the desired and summed signals. The combining process is not a summing process, but rather it coalesces signal samples into the same signal sample (coefficient) record for output onto a single port.

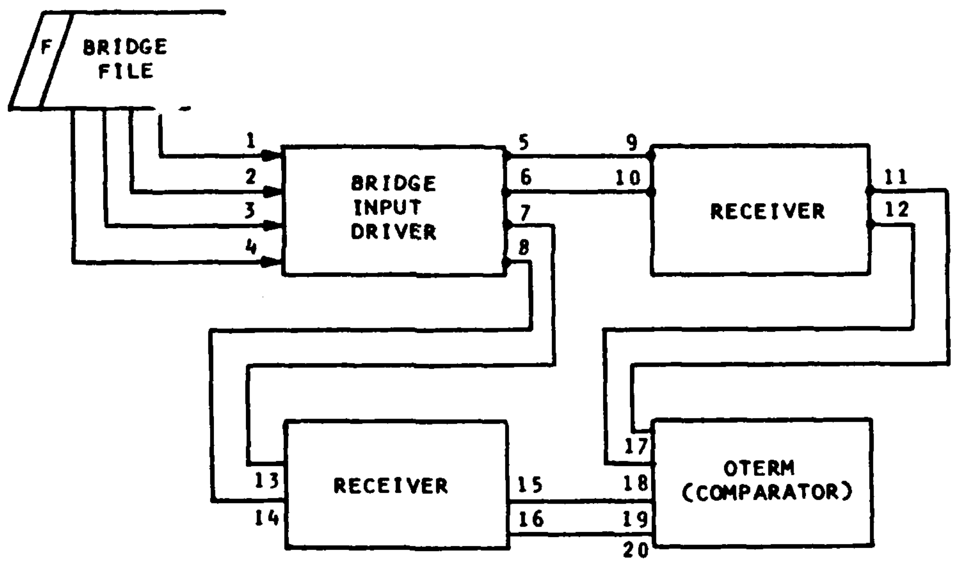


Figure 3. A TSM in which the "desired" signal is read directly from the Bridge File.

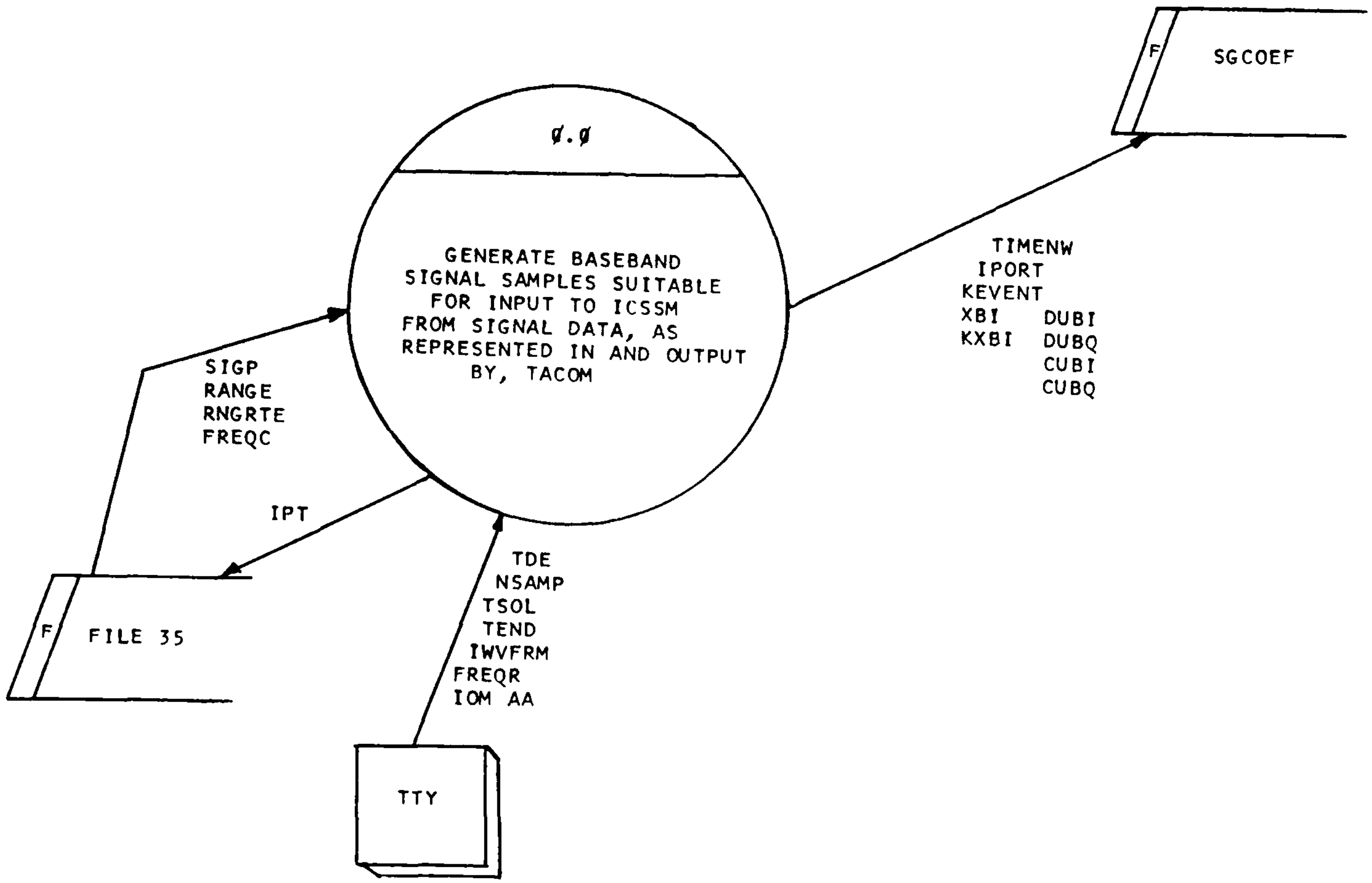


Figure 4. Bubble Charts for TACOM/ICSSM Bridge

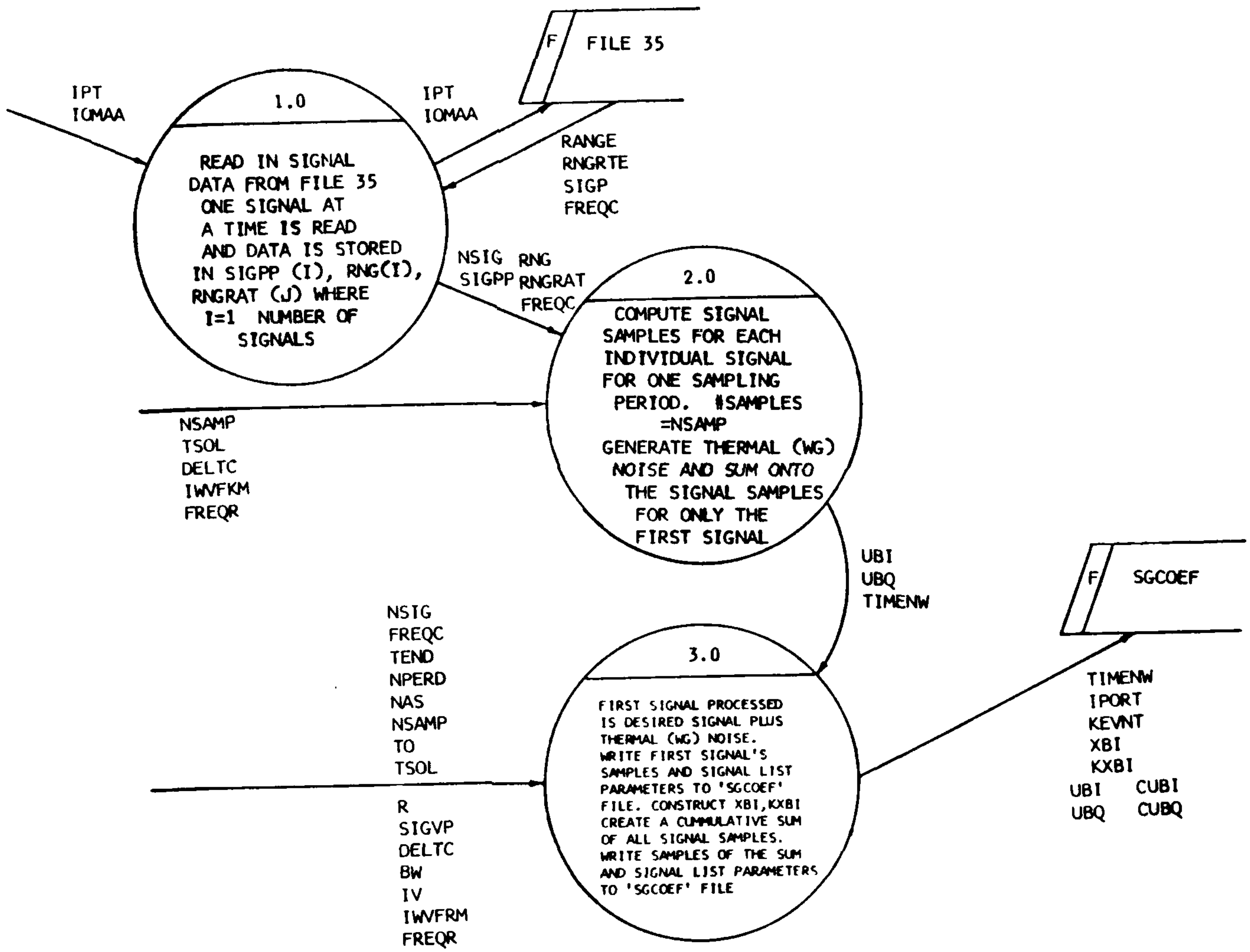


Figure 5. Bubble Charts for TACOM/ICSSM Bridge

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

END

FILMED

1-84

DTIC